SNOMED International

Leading healthcare terminology, worldwide

SNOMED CT
Terminology Services Guide

Publication date: 2020-09-30

Web version link: http://snomed.org/tsg

SNOMED CT document library: http://snomed.org/doc

This PDF document was generated from the web version on the publication date shown above. Any changes made to the web pages since that date will not appear in the PDF. See the web version of this document for recent updates.

SNOMED International

Leading healthcare terminology, worldwide

Table of Contents

1 Introduction....2
2 Terminology Services Overview....4
- 2.1 Terminology Services Definition....4
- 2.2 Terminology Service Categories....4
- 2.3 Terminology Service Roles....8

3 Terminology Service Use Cases....18
- 3.1 Explore and Review SNOMED CT....18
- 3.2 Support EHR Data Entry....19
- 3.3 Display EHR Data....27
- 3.4 EHR Reporting and Analytics....29
- 3.5 Reference Set Editing....30
- 3.6 Apply Mapping Reference Sets....32
- 3.7 Terminology Change Management....34
- 3.8 Support Terminology Authoring and Review....48

4 Terminology Service Types....50
- 4.1 Select Edition and Version....50
- 4.2 Get a Concept, Description or Relationship....56
- 4.3 Get Terms for a Concept....62
- 4.4 Get Definition of a Concept....73
- 4.5 Get and Test Concept Subtypes and Supertypes....82
- 4.6 Get and Test Reference Set Membership....94
- 4.7 Validate and Apply Expression Constraints....101
- 4.8 Find Concepts....115
- 4.9 Identify Changes to the Terminology....129
- 4.10 Get Data from a Reference Set....134
- 4.11 Get History Data....140
- 4.12 Get Mapping Data....149
- 4.13 Get Concept Model Rules....162
- 4.14 Validate Concept Definitions and Expressions....177
- 4.15 Test Expression Subsumption....183

5 Service Implementation Considerations....186
- 5.1 Terminology Data Storage and Access....186
- 5.2 Enabling Access to Editions and Versions....187
- 5.3 Release Types and Versioned Views....192

5.4 Importing Release File Data ... 203
Appendix A: Summary of Types and Use Cases ... 207
Appendix B: Editions, Versions and Extensions ... 211
Footnotes ... 214

The Terminology Services Guide describes the terminology services required to support effective access to the content and features of SNOEMD CT. This guide primarily focuses on the terminology service requirements of applications that enable the use of SNOMED CT for clinical purposes.

Web browsable version: http://snomed.org/tsg

SNOMED CT Document Library: http://snomed.org/doc

1 Introduction

Background

SNOMED CT is a clinically validated, semantically rich, controlled terminology. SNOMED CT is comprised of concepts with human-readable descriptions and a hierarchical structure supported by machine-readable logic-based definitions. SNOMED CT is used within electronic health records to support data capture, retrieval, and subsequent analysis and reuse for a range of purposes. The SNOMED CT design also includes features that support multilingual terms, subsets, maps to other codes systems, formal constraint specifications and comprehensive history tracking.

When implemented in software applications, SNOMED CT can represent clinically relevant information consistently, reliably and comprehensively as an integral part of electronic health records. This requires effective access to terminology content in ways that leverage the features of the terminology. A terminology service is a software function that interfaces with and provides access to information from one or more representations of a terminology.

Some services described in this guide are also applicable to other code systems, classifications and terminologies. However, delivery of the full set of benefits of SNOMED CT also requires terminology services that access and make effective use of features that are specific to SNOMED CT.

Purpose

The purpose of this document is to identify and describe the terminology services required to support effective access to the content and features of SNOMED CT. The guide also outlines approaches to delivery for some of the most important services. In some cases, more extensive details of service requirements and delivery options are provided by reference to other SNOMED CT documents or appendixes to this document. Where appropriate, this document may also include practical illustrations of a service by reference to specifications of existing tools or interfaces.

Scope

The content of this guide is primarily focused on the terminology service requirements of applications that enable the use of SNOMED CT for clinical purposes. This include electronic health record applications that enable entry, viewing, and reporting of clinical data. It also includes applications that support clinical data analytics and decision support. Some services and use cases mentioned in the guide are also relevant to terminology content authoring. However, it is important to note that the scope of this guide does **not** cover the full set of requirements for terminology development, authoring, and distribution services.

Disclaimer

Specific references to terminology service tools and interfaces are provided as illustrative examples only. For the avoidance of misunderstanding, these illustrative examples are not recommendations. Referenced tools and the interfaces they support may be subject to license conditions, charges, and changes outside the control of those maintaining this document. Therefore, if any examples provided in this document do not appear to work, please check the documentation provided by the licensor of the tool or interface.

Audience

This guide is designed to be of value and interest to the following audience groups:

- **Consumers of SNOMED CT terminology services**:
    - ***Procurers and implementers of healthcare applications***
        - People involved in procuring and/or deploying applications that enable entry, viewing, reporting, and analysis of healthcare data coded using SNOMED CT

        - ***Testers and evaluators of specific healthcare applications***
            - End-users responsible for testing or evaluating applications that enable entry, viewing, reporting, and analysis of healthcare data coded using SNOMED CT
- **Providers of healthcare applications that use SNOMED CT:**
    - People involved in designing, developing, or maintaining applications that enable entry, viewing, reporting, and analysis of healthcare data coded using SNOMED CT
- **Providers of SNOMED CT terminology services**:
    - People involved in designing, developing, maintaining, or delivery of terminology services that include support of access to SNOMED CT
    - The needs of each of these audience groups vary, and as a result, the sections likely to be of most interests to each of these audience groups are noted in the *Document Structure* section below

Document Structure

This document is organized into five main chapters:

- Chapter 2 introduces the definition of SNOMED CT terminology services, it describes the general terminology service categories, and it presents three key roles with an interest in SNOMED CT terminology services and provides notes on issues relevant to each role.
- Chapter 3 outlines the practical use cases for terminology services.
- Chapter 4 describes the different types of terminology services required to enable effective use of SNOMED CT.
- Chapter 5 provides practical information regarding the use of the SNOMED CT release files to meet specific terminology service requirements.

2 Terminology Services Overview

This chapter defines SNOMED CT terminology services, and it describes the general terminology service categories. Furthermore, it presents three key roles with an interest in SNOMED CT terminology services including end-users and those involved in the procurement of healthcare applications.

2.1 Terminology Services Definition

SNOMED CT terminology services are defined as a set of software functions that interface with and provide effective access to the content and features of one or more SNOMED CT editions.

Notes

- Each terminology service is a software function that interfaces with and provides access to information from one or more representations of a terminology.
- The software application that provides *SNOMED CT terminology services* is known as a SNOMED CT terminology server.
- While terminology services may be applicable to several different terminologies, applications that provide *SNOMED CT terminology services,* should support access to the specific design features of SNOMED CT. These include the subtype hierarchy, description logic concept definitions, reference sets and expression constraints.

Examples

- Find a concept based on a set of search criteria. The criteria may include text to be matched against terms associated with the concept and constraints requiring a concept to be a subtype of an identified *concept* or a member of an identified reference set.
- Show the preferred term associated with an identified concept.

Related Links

- SNOMED CT terminology server
- Terminology service

This page includes the full text of an entry in the SNOMED CT Glossary.

2.2 Terminology Service Categories

SNOMED CT terminology services can be subdivided into categories based on the following two defining characteristics:

1. **Access requirements**: Does the service need to update the terminology?
2. **User interface requirements**: Does the service include its own user interface controls?

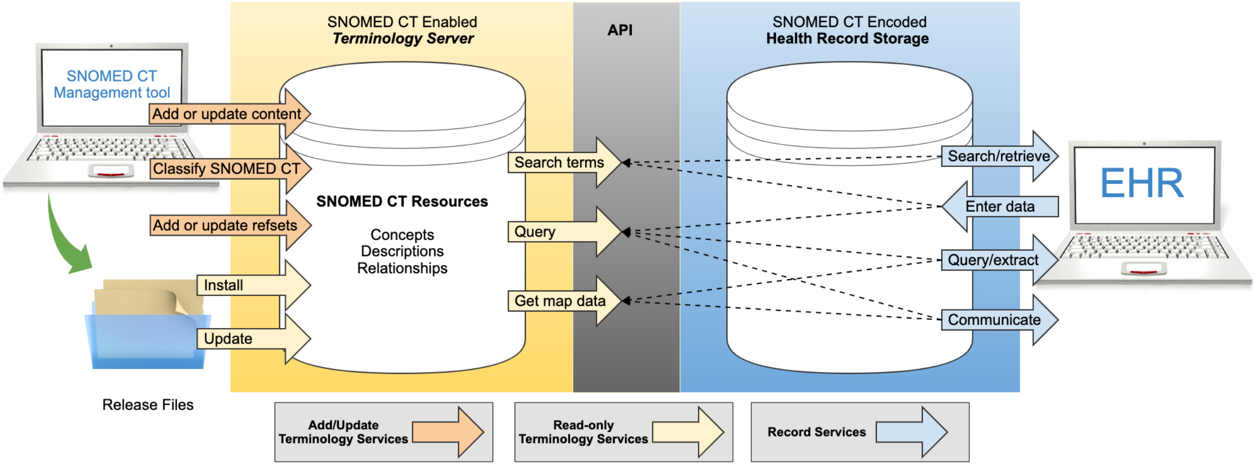
Access Requirements

Table 2.2-1 describes the distinction between terminology services that provide read-only access to the terminology and those that also allow the terminology to be updated. Practical requirements for using SNOMED CT to enter, display, and report clinical data can be met by read-only terminology services. Services that are able to update the terminology are only required by those involved in the development, maintenance, or customization of the terminology. Figure 2.2-1 illustrates the association between the different types of terminology services and record services.

Table 2.2-1: Terminology Access Requirements

Characteristic	Description	Additional Notes
Read-Only	Read-only terminology services enable access to SNOMED CT content and features.	These services meet requirements for practical use of SNOMED CT including collecting, displaying, communicating, and analyzing SNOMED CT coded data.
Add-Update	Add-update terminology services can add, modify, or inactivate SNOMED CT components and/or reference set members.	These services include functions that support terminology authoring, maintenance, and distribution. A full suite of development services meets the requirements of organizations responsible for creating and maintaining a SNOMED CT edition, or a SNOMED CT extension containing additional clinical concepts. Limited sets of development services can meet the requirements of organizations responsible for a SNOMED CT extension that consists only of reference sets representing subsets, maps, or data that is used to customize the terminology to meet specific purposes.

Figure 2.2-1: Examples of Terminology services and Record services and their associations.



User Interface Requirements

Table 2.2-2 differentiates between services based on whether the service includes user interface components, in addition to its application programming interface. Terminology services that do not include user interface components are capable of delivering a full range of essential terminology services but they require the client application to provide the user interface components to interact with those services. Terminology services that include user interface components have the potential to simplify client application development and configuration. Some examples of potentially beneficial terminology services with user interfaces are noted in Table 2.2-2. Section 3 Terminology Service Use Cases notes the need for combinations of the terminology services identified in 4 Terminology Service Types to address each use case. Most of these use cases also need interfaces to allow users to interact and access the results of those services. However, the detailed specification of the user interface functionality of these combined services is beyond the scope of the current version of this guide[1].

Table 2.2-2: User Interface Inclusion

Characteristic	Description	Additional Notes
No UI	Terminology services that only provide access to SNOMED CT through an API.	Client applications using these services are responsible for providing any user interfaces required to enable practical use of these services. Client developed user interfaces can be closely integrated with the look and feel of the client application user interface. They limit dependency on a specific terminology service provider as services without a user interface have less variability and are more likely to share include common features.

UI	Terminology services that, in addition to an API, also provide a user interface through which users can interact with the terminology.	Services in this category range from individual terminology bound user interface controls to fully functioning tools that enable viewing or editing the terminology. User interface controls included as part of terminology service may facilitate more rapid development. They may be particularly useful in cases where a client application has limited requirements for searching and displaying SNOMED CT content. **Examples:** • Terminology search control supporting: (Use Cases 3.2 Support EHR Data Entry) ▪ Text searches with search constraints set by data entry. ▪ Display of results in a way that facilitates the selection of an appropriate concept. ▪ Option to enable the addition of postcoordinated refinements. • Report and analysis query development tool: (Use Cases 3.4 EHR Reporting and Analytics) ▪ Enabling creation of valid expression constraints or SNOMED CT queries.

Examples

Table 2.2-3 provides examples of services in each of the four categories defined by applying the terminology access and user interface criteria identified in 2.2 Terminology Service Categories.

Table 2.2-3: Terminology Service Category Examples

Terminology Access	User Interface	Examples
Read-Only	**No UI**	Get details of a concept with a specified concept id.
		Get sets of concepts that match term search criteria and/or expression constraints.
		Get reference set data related to a concept or description. Including, subset membership, description acceptability in a specified language, maps to other terminologies, and access to terminology history data.
		Test which of a set of concepts and/or expressions is subsumed by a specified concept or expression constraint.
	UI	A SNOMED CT browser that enables exploration of the terminology and provides an API through which an application may set or read the results of user actions in the user interface.
		a terminology bound user interface control is a list control that can be populated with a list of SNOMED CT concepts specified by membership of a reference set or conformance with an expression constraint. The API enables the list content criteria to be set and also allow user selections to be read.
		Software tools that provide a user interface through which to analyze records that include SNOMED CT coded data.
Add-Update	**No UI**	Create a new concept.
		Add or modify axioms to enhance the definition of the concept.
		Classify SNOMED CT content to add inferences based on stated defining axioms.
		Add descriptions to a concept.
		Inactivate a concept or description.
		Create a new reference set of a particular type.

		Add members to a reference set.
	UI	SNOMED CT authoring tools that support the creation of concepts with appropriate sets of descriptions and formal axiom-based definitions.
		Tools that support the development and maintenance of subsets of concepts represented as simple reference sets.
		Tools that support the creation and maintenance of SNOMED CT language translations, by enabling descriptions to be added and assigned acceptability settings in a language reference set.
		Tools that support the development of maps between SNOMED CT and other terminologies, classifications, or code systems by enabling creation and maintenance of mapping reference sets.

Footnotes

[1] Many of the general use cases identified in 3 Terminology Service Use Cases can be met in different ways by different combinations of the terminology services detailed in 4 Terminology Service Types with user interface components or forms. Therefore, detailed specifications of the specific functionality of combined terminology services would inevitably be either incomplete or overly restrictive. Furthermore, user interfaces presented by a combined terminology service will typically need to be integrated with client applications. Client applications may adopt different user interface styles and these styles may evolve overtime. Therefore, flexibility in the design of the user interface through which a service is accessed may be preferable to a rigid detailed specification of each type of service. Depending on feedback from readers, consideration will be given to providing more guidance on combined terminology services in future versions of this guide.

2.3 Terminology Service Roles

This section identifies three roles with an interest in SNOMED CT terminology services. The perspective of each of these roles is summarized with notes on general issues that those in particular roles should consider and references to relevant sections in the guide. The roles considered are as follows:

2.3.1 Terminology Service Users

Terminology Server Users include

- End-Users
- People involved in procurement

End-Users

The end users of SNOMED CT terminology services include healthcare professionals and other people who enter, view, report, and analyze healthcare data coded using SNOMED CT. Terminology services may sound like a technical matter of little interest to users but the quality and performance of these services can make a big difference to the usability of a healthcare application. Poorly implemented terminology services can make it hard to locate appropriate concepts, distracting, and frustrating health professionals as they struggle to enter notes. The end result is likely to be incomplete records or inaccurate recording due to the inability to find the correct concept. In contrast, well-designed terminology services used effectively by healthcare applications facilitate rapid and accurate entry of clinical data. This not only assists the delivery of patient care but also provides a valuable source of information for communication, reporting, and valuable analysis.

Terminology services that enable rapid searches for concepts in relevant areas of the SNOMED CT hierarchy are a key factor in facilitating accurate data entry and minimizing the risk of errors. When creating task-oriented forms or templates for data entry it is even more important to use search techniques that pinpoint the correct concepts, because any errors in a template will inevitably be duplicated by records created using that template.

Terminology services also need to support data retrieval for retrieval, analysis, and decision support. This requires terminology services that can examine the definition of a concept and determine if it meets the criteria specified in a query or expression constraint. A terminology service that supports rapid evaluation of these types of queries can be a key factor in facilitating the delivery of essential reports and valuable analytics.

Most end-users are unlikely to have a direct role in the development or selection of terminology services. However, those who are involved in designing, procuring, and implementing these services should recognize the importance of ensuring a positive end-user experience for people whose day-to-day work requires frequent access to SNOMED CT.

Procurement

Evaluating the Usability of Terminology Services

Organizations procuring applications that require access to SNOMED CT should carefully evaluate the way that the application provides access to terminology content and features. This evaluation should consider practical usability and also the flexibility of the underlying design. It should also consider the context of the existing technical architecture, as this will drive the type of service being procured.

To facilitate the assessment of the way an application interacts with SNOMED CT, the following steps are recommended:

1. Identify user activities that involve interaction with the terminology
    - It may be helpful to refer to the list of use cases in Section 3.2 to assist the identification of specific activities that are relevant to the application being procured
2. Identify groups of users that regularly undertake each of the identified activities
3. Request representatives of the affected groups of users to assist with the evaluation of those activities
    - Those involved in this evaluation may find it useful to refer to the use cases in Section 3.2 and the specific service requirements in Section 3.1 which may suggest a particular factor underlying any issues they experience

Evaluating Application Options for Access to Terminology Services

Another important point to consider when procuring a solution that requires access to SNOMED CT. is the way (or ways) in which the solution provides access to the terminology. A solution may include built-in terminology services, it may provide an interface to terminology services from a specific provider, or it may offer options to use terminology services from different providers who support a common interface.   Table 2.3.1-1 outlines the advantages and disadvantages of each of these options from the perspective of a procuring organization.

Table 2.3.1-1: Options for Application Access to Terminology Services - Procuring Organization Perspective

Terminology Services Options	Advantages for Procuring Organization	Disadvantages for Procuring Organization

Application includes built-in functions for the terminology services it requires	• Integrated solution without external dependencies • Potential for optimization of specific terminology services needed by the application • Single source for support	• Dependent on the quality of the application provider's solution and its future maintenance • No option to select an enhanced set of terminology services from a specialist provider • Risk of non-alignment between SNOMED CT editions/versions in terminology services used by different applications
Application uses terminology services provided by the same organization but independent from the application	• Purpose-built solution matched to the application • Single source for support • Potential for reuse of terminology services by other applications	• Dependent on the quality of the application provider's terminology services and its future maintenance • No option to select an enhanced set of terminology services from a specialist provider • Option for other applications to use the services is limited to those able to use the interface provided by the application vendor • Risk of non-alignment between SNOMED CT editions/versions in terminology services used by different applications
Application uses terminology services provided by a specified third party	• Application provider is responsible for maintaining compatibility of their application with the chosen terminology services interface. • Potential for reuse of terminology services by other applications	• Dependent on the quality of the specified terminology services and its future maintenance • Option for other applications to use the services is limited to those able to use the interface provided by the chosen terminology services • Risk of non-alignment between SNOMED CT editions/versions in terminology services used by different applications

Application supports use of terminology services that offer an interface that conforms to a published specification[1]	• Application provider is responsible for maintaining compatibility of their application with the chosen terminology services interface. • Choice of terminology services may allow reuse of terminology services already used in the organization • Choice of terminology services may allow migration to enhanced terminology services offered by a different provider • Future options for reuse of terminology services by other applications that support the same published interface specification • Future options to switch to another provider of terminology services that conform to the same published interface specification	• Possible differences in performance, functionality, and results between different implem entations of a chosen terminology services interface • Separate support arrangement for terminology services may make responsibility for errors or performance issues less clear • Option for other applications to use the services may be limited by the chosen interface specification
Application configurable to support use of a specified range of terminology services	• Application provider is responsible for maintaining compatibility of their application with the chosen terminology services interfaces. • Choice of terminology services may allow reuse of terminology services already used in the organization • Choice of terminology services may allow migration to enhanced terminology services offered by a different provider • Future options for reuse of terminology services by other applications that support the same published interface specification • Future options to switch to another provider of terminology services that conform to the same published interface specification	• Separate support arrangement for terminology services may make responsibility for errors or performance issues less clear • Option for other applications to use the services may be limited by the chosen interface specification

[1] Examples of defined terminology services interfaces include the Snowstorm API and the HL7 FHIR terminology server API.

2.3.2 Healthcare Application Providers

Addressing Terminology Service Requirements

Application providers should refer to the following sections of this guide for statements of terminology service requirements that need to be met for effective access to SNOMED CT.

- 2.3.1 Terminology Service Users
    - Which notes user requirements for high performance services that provide appropriate access to SNOMED CT.
- 4 Terminology Service Types
    - Which describes specific terminology services or functions that are required to enable effective use of SNOMED CT
- 3 Terminology Service Use Cases
    - Which provides examples of practical use cases involving access to SNOMED CT that an application may need to complete. For each of these use cases it identifies one or more of the required terminology services that can be used to complete the required activity.

As noted in the following section, these requirements can be met either directly by the application or by use of SNOMED CT terminology services accessed through an API.

Application Options for Access to Terminology Services

Applications that require access to SNOMED CT are direct users of SNOMED CT terminology services. Organizations that design, develop, supply and support those applications can enable access to SNOMED CT in several different ways. The main options are identified in Table 2.3.2-1 with notes on the advantages and disadvantages of each option from the perspective of the application provider.

The first two options shown in Table 2.3.2-1 require the application provider to also develop and maintain terminology services. Application providers considering those options should also take a look at the notes on the Terminology Services Provider Role.

Table 2.3.2-1: Options for Application Access to Terminology Services - Application Provider Perspective

Terminology Services Options	Advantages for Application Provider	Disadvantages for Application Provider
Application includes built-in functions for the terminology services it requires	• Integrated approach may simplify installation. • No external dependency on third parties for supply or support of terminology service.	• Application provider must create, maintain and support terminology services. • Application provider must keep pace with any significant changes to the terminology design that impact application user requirements. • Application provider must enable each customers with access to an appropriate version of the SNOMED CT edition that they require. • Application may be less attractive to organizations already using a terminology server.
Application uses a defined interface to terminology services provided by the same organization	• Purpose built terminology server matching to application requirements. • No external dependency on third parties for supply or support of terminology service. • Potential for terminology services to be licensed as a product for use with other applications.	• Application provider must create, maintain and support terminology services. • Application provider must keep pace with any significant changes to the terminology design that impact application user requirements. • Application provider must enable each customers with access to an appropriate version of the SNOMED CT edition that they require. • Application may be less attractive to organizations already using a different terminology server.

Application uses terminology services provided by a specified third party	• No need to create, maintain and support terminology services. • Less important to keep up with terminology developments (as terminology server should manage this). • Less responsibility for ensuring customers have access to an appropriate version of the SNOMED CT edition that they require (as terminology server should manage this).	• Application provider must maintain and support application calls to interface with chosen third party terminology services. • External dependency on supplier of chosen terminology services for support. • Risk of support conflicts where source of error may be the terminology services or the application • Risk of limitations to development if the chosen terminology server does not update to take account of changes to SNOMED CT or additional terminology access requirements of the application. • Application may be less attractive to organizations already using a different terminology server.
Application supports use of terminology services that offer an interface that conforms to a published specification[1]	• No need to create, maintain and support terminology services. • Less important to keep up with terminology developments (as terminology server should manage this). • Less responsibility for ensuring customers have access to an appropriate version of the SNOMED CT edition that they require (as terminology server should manage this). • Application may be more attractive to organizations already using the supported terminology service interface. • Opportunity for customers to migrate to a different source of terminology services if their current terminology provider is not meeting their requirements or ceases to develop or support their service.	• Application provider must maintain and support application calls to interface with third party terminology services that implement the chosen interface. • Risk that different implementations of terminology services using the same interface may have different performance characteristics. • Risk that different implementations of terminology services using the same interface may return different results in response to some service requests. • External dependency on chosen terminology service providers for support. • Risk of support conflicts where source of error may be the terminology services or the application • Risk of limitations to development if the chosen terminology server does not update to take account of changes to SNOMED CT or additional terminology access requirements of the application.

Application configurable to support use of a specified range of terminology services	• No need to create, maintain and support terminology services. • Less important to keep up with terminology developments (as terminology server should manage this). • Less responsibility for ensuring customers have access to an appropriate version of the SNOMED CT edition that they require (as terminology server should manage this). • Application may be more attractive to organizations already using the one of the supported terminology services. • Opportunity for customers to migrate to a different source of terminology services if their current terminology provider is not meeting their requirements or ceases to develop or support their service.	• Application provider must maintain and support application calls to interface with third party terminology services that implement the chosen terminology services interface. • Risk that different supported terminology services may return different results in response to some service requests. • Risk that different supported terminology services may return different results in response to some service requests. • External dependency on chosen terminology server for support. • Risk of support conflicts where source of error may be the terminology server or the application • Risk of limitations to development if the chosen terminology server does not update to take account of changes to SNOMED CT or additional terminology access requirements of the application.

[1] Examples of defined terminology services interface include the Snowstorm API and the HL7 FHIR terminology server API.

2.3.3 Terminology Service Providers

Terminology Service Requirements

Terminology service providers should refer to the following sections of this guide for statements of requirements that their services should meet when accessing SNOMED CT:

- 2.3.1 Terminology Service Users
    - Outlines user requirements for high-performance services that provide appropriate access to SNOMED CT.
- 2.3.2 Healthcare Application Providers
    - Outlines requirements from the perspective of healthcare applications that consume terminology services.
- 3 Terminology Service Use Cases
    - Provides examples of practical use cases involving access to SNOMED CT that an application may need to complete. For each of these use cases, it identifies one or more of the required terminology services that can be used to complete the required activity.
- 4 Terminology Service Types
    - Describes specific terminology services or functions that are required to enable effective use of SNOMED CT.

Support for Different Terminologies and Interfaces

Clinical systems can be effectively implemented using a service oriented architecture, in which the terminology services are designed as an independent software component, accessible via an API (Application Programming Interface) gateway. This enables the other system components to access the key services without being affected by changes to the way these services are implemented.

When designing terminology services for SNOMED CT, it is important to utilize the sophisticated design features of the terminology, as this will support its effective use. When designing terminology services to support a range of different terminologies, it is important to consider the commonality between these terminologies. Table 2.3.3-1 summarizes some of the advantages and disadvantages of designing terminology services that only work with SNOMED CT, compared with designing services that can also enable access to terminologies or code systems. It also identifies advantages and disadvantages of supporting one or more interfaces through which applications can access SNOMED CT and/or other terminologies.

Table 2.3.3-1: Terminology Service Design Options

Options		**Notes**
Terminologies Supported	**SNOMED CT only**	A terminology service provider may develop and deliver services that are specific to SNOMED CT. • The advantage of this approach is that it allows services to be tailored to the structure of and features of SNOMED CT. • The disadvantage is that it may make the services less attractive to organizations looking for a more general terminology services solution.
	SNOMED CT and other code systems	A terminology service provider may develop and deliver services that support access to SNOMED CT and other terminologies, code systems, or classifications. • The advantage of this approach is that it provides a single solution for customers that require access to a range of different terminologies. • The disadvantage is that there is a risk that the services offered may not make full use of features that are specific to SNOMED CT and the other code systems.
Terminology Service Interfaces Supported	**Proprietary interface**	A terminology service provider may specify their own interface through which client applications access SNOMED CT and/or other terminologies. • The advantage of this approach is that the interface can be designed to take account of special features that the developer has included. • The major disadvantages of a proprietary interface is that ▪ application providers may not be willing to invest effort in supporting different proprietary interfaces ▪ consumers get tied-in to a specific provider, because migration to a different, and potentially better, provider is too complicated
	SNOMED CT specific interface	A terminology service provider may support an interface that is specifically designed to work with SNOMED CT. For example, SNOMED International's own terminology server Snowstorm provides an interface that is specifically designed to work with SNOMED CT. • The advantage of this approach is that it allows optimization of services that are specific to the design and features of SNOMED CT. • The disadvantage is that it requires application providers to use a specific interface for accessing SNOMED CT while they need to use other interfaces to access other code systems.

	A general-purpose terminology service interface	A terminology service provider may support a general-purpose interface that can be used to access a wide range of different terminologies, code systems, and classifications. For example, the FHIR Terminology Service (part of the HL7® FHIR® Specification[1]) defines a general-purpose terminology interface applicable to a wide range of healthcare terminologies and code systems. • One advantage of this approach is that it allows an application provider to use the same terminology service interface to access SNOMED CT and other code systems to which they require access. Another advantage is that an application provider can switch between different implementations of terminology services that support the same interface. • The disadvantage of a general-purpose interface is that it may not facilitate optimized access to some of the specific features of SNOMED CT.
	Multiple terminology service interfaces	A terminology service provider may support several terminology service interfaces. For example, SNOMED International's Snowstorm server supports both the Snowstorm REST API and the FHIR Terminology Services API. • The main advantage of this approach is that it combines the advantages of a SNOMED CT specific interface and the advantages of a general-purpose interface. It allows a general-purpose interface to be used to access other code systems, while also supporting the optimization of services that address SNOMED CT specific requirements. • The main disadvantage of this approach is that it requires the service provider to maintain alternative ways to access similar services. If a terminology service supports more than one interface specification, it is important to document any differences in results or performance. A service that optimizes access through a proprietary interface may not deliver the same performance when accessed through a more general interface. In some cases, the output of similar services may differ due to differences in the specifications or due to limitations of a particular implementation.

Footnotes

[1] HL7® and FHIR® are both registered trademarks of HL7 (www.hl7.org).

3 Terminology Service Use Cases

This section describes practical use cases which require the use of specific terminology services.

3.1 Explore and Review SNOMED CT

SNOMED CT browsers and other applications that enable access SNOMED CT concepts allow the content and structure of the terminology to be explored. This exploration can serve many different purposes, ranging from allowing people to get a broad idea of the scope of SNOMED CT to looking for specific concepts that interest them. The same applications can be used to enable a more thorough review of the terminology in a particular specialty area to assess the breadth, depth and applicability of SNOMED CT to a particular set of requirements. Tools that enable visual review of a released SNOMED CT Edition are also valuable for quality assurance and when considering requests for addition or changes to the terminology.

Table 3.1-1 shows a summary of the terminology services required to support effective exploration and review of SNOMED CT content.

Table 3.1-1: Terminology Services Required to Support Exploring and Reviewing SNOMED CT

Practical Requirement	**Status** [1]	**Required Terminology Services** [2]	**Additional Terminology Service Dependencies** [3]
Enable the selection of SNOMED CT edition and version to be explored.	REQUIRED	4.1 Select Edition and Version	N/A
Enable concepts to be found by identifier.	REQUIRED	4.2 Get a Concept, Description or Relationship	N/A
Enable the display of terms for any identified concept.	REQUIRED	4.3 Get Terms for a Concept	N/A
Enable the display of the definition of a concept.	REQUIRED	4.4 Get Definition of a Concept	N/A
Enable the display of the SNOMED CT subtype hierarchy, including supertype parents and subtype children.	REQUIRED	4.5 Get and Test Concept Subtypes and Supertypes • Get subtype children of a concept • Get supertype parents of a concept	N/A
Enable concepts to be found by term searches.	REQUIRED	4.8 Find Concepts • Find concepts by term search only.	N/A

Enable concepts to be found by constrained term searches.	OPTIONAL	4.8 Find Concepts - Find concepts by constrained term search.	One or more of the following depending on range of search constraint types supported 3.1.5 Get and Test Concept Subtypes and Supertypes - Test subsumption between two concepts; and/or - Test a set of concepts for subsumption 3.1.6 Test Reference Set Membership 3.1.7 Apply Expression Constraints
Enable the display of an indication of the reference sets of which a specified concept or description is a member.	OPTIONAL	3.1.6 Test Reference Set Membership	N/A
Enable the display of the acceptability of a specified description in an identified language reference set	OPTIONAL	4.10 Get Data from a Reference Set	N/A
Enable the display of reference set data related to a specified reference set member.	OPTIONAL	4.10 Get Data from a Reference Set	N/A
Enable the display of all members of a specified reference set and the reference set data related to each member.	OPTIONAL	4.10 Get Data from a Reference Set	N/A

Footnotes

[1] Applications designed to address this use case must support the practical requirements marked as *Required*. Support for the practical requirements marked as *Optional* is recommended as these provide enhanced functionality that may be required by some users.

[2] In most cases, a reference to a subsection of 4 Terminology Service Types, implies a requirement for all services marked as *Required* in that subsection. However, where a reference is followed by a bulleted list, that list specifies the specific terminology services required. Some of the specific services listed as required for an *Optional* practical requirement may be marked as *Recommended* in the referenced subsection.

[3] The *Additional Terminology Service Dependencies* column contains references to services on which a *Required Terminology Service* depends. This column does not restate dependencies on services listed as required service or additional dependencies for essential requirements listed in earlier rows. A full list of the dependencies of each terminology service is provided in the relevant subsection of 4 Terminology Service Types.

3.2 Support EHR Data Entry

An electronic health record (EHR) that represents clinical information using SNOMED CT must support efficient entry of data accurately coded as SNOMED CT concepts and/or expressions. Data entry approaches need to be tailored to fit the needs of different healthcare professionals in a variety of clinical situations. However, the end results of different data entry techniques must be record entries that are consistent and comparable. The following sections outline the ways in which terminology services can be used to support the design and application of effective data entry techniques.

3.2.1 EHR Data Entry Overview

This section provides a general introduction to the terminology service requirements related to EHR data entry. It outlines the rationale for the following sections on data entry design and practical data entry.

Background Reading

Readers of this section of the guide are advised to read the following sections of the SNOMED CT Search and Data Entry Guide.

- 4. Optimizing Searches - describes effective ways to optimize SNOMED CT searches.
- 5. Optimize Display of Search Results - provides guidance on ordering and structuring lists of search results.
- 6. Data Entry - describes a range of techniques that facilitate the effective recording of SNOMED CT concepts or expressions in health records.

Data Entry Context

Entry of data into an Electronic Health Record (EHR) requires access to services that enable the user to rapidly locate and select the concepts and terms that need to recorded. There are a wide range of different EHR data entry scenarios determined by the type of healthcare encounter and the reason for that encounter. A typical data entry scenario involves recording data in different data entry contexts. The data entry contexts are typically distinguished by section headings (e.g. "Surgical History"), individual data item labels (e.g. "Initial Diagnosis") and in some case by specific questions with a limited range of answers (e.g. "Family history of heart disease?" with values "Yes", "No" or "Unknown").

Data entry design needs to take account of data entry context for two reasons.

1. The range of concepts that can rationally be entered is determined by the data entry context.
    - Table 3.2.1-1 illustrates this point with examples of three data entry contexts. In each of these contexts, there are constraints on the range of SNOMED CT concepts that it is rational to enter.
2. The interpretation of concepts or other data entered may be affected by the data entry context.
    - Table 3.2.1-1 provides examples of ways in which the data entry context can affect the interpretation of recorded data. To ensure appropriate interpretation, relevant information about the data entry context must be stored with or linked to the data entered.

Table 3.2.1-1: Data Entry Context Examples

Data Entry Context	Constraints on Values	Interpretation of Recorded Data

Initial diagnosis on an encounter or admission form	A concept recorded in this data entry context should be a subtype of 64572001 |Disease (disorder)|. **Example** A healthcare professional assesses an unconscious person in the emergency room and concludes that the patient is suffering from effects of alcohol. To enter this initial diagnosis they type "alcohol". An unconstrained search for "alcohol" will find a concept with the term |Alcohol| and may show that at the top of a search list. The problem is that this term is a synonym for the 53041004 |Alcohol (substance)|. While a substance may be relevant to the patient's condition a substance is not a valid diagnosis. A search for the term ""alcohol" constrained to subtypes of 64572001 |Disease (disorder)| avoids this error and results a shorter, more appropriate list including concepts such as 25702006 |Alcohol intoxication|.	The fact that this is an "Initial diagnosis" data entry context needs to be captured so that the record can be appropriately interpreted. **Note** Subsequent investigations and assessments may refute the initial diagnosis. In this case, the initial diagnosis should not be presented as a condition the patient has actually had. However, it may remain relevant as the rationale for initial treatment and investigation. Furthermore, a subsequently refuted initial diagnosis may also be of interest from the perspective or service administration, clinical audit and research.
Surgical history as part of past medical history	A concept recorded in this data entry context could either be: • A subtype of 161615003 |History of surgery (situation)|; or • A subtype of 387713003 |Surgical procedure (procedure)| with an indication that this is past history. **Example** A past history of an appendectomy could be recorded in different ways including: • Using the concept 428251008 |history of appendectomy|; or • Using the concept 80146002 |appendectomy| with an indication that this was reported by the patient as having been done in the past.	The "Past History" data entry context needs to be captured so that the record can be appropriately interpreted. It must be possible to distinguish a past history record of a procedure from a contemporaneous record of the same procedure. **Note** The definitions of concepts that are subtypes of 161615003 |History of surgery (situation)| formally represent the past history context by including the attribute 408731000 |Temporal context| = 410513005 |In the past|. If subtypes of 387713003 |Surgical procedure (procedure)| are to be used to record surgical history, these record entries must include an explicit indication of the past history context. In either case, it may also be useful to indicate the source of the information (e.g. reported by patient, derived from original record) and actual or approximate date of the procedure.

Symptom check list with yes or no options for each symptom	Each question should be bound to a concept that represents the relevant symptom. These concepts should be subtypes of 404684003 Clinical finding. The simplest way to represent the "yes" and "no" answers to a question like this is to record the relevant finding concept if the answer is "yes" and not to create a record if the answer is "no". However, where the answer "no" has clinical significance, alternative approaches discussed in the next column may be preferable. **Example** A question about whether a patient has a sore throat would be linked to the concept 162397003 |sore throat|. - If the patient answers "yes" then 162397003 |sore throat| is added to the patient's record. - If the patient answers "no" then no entry is added to the record (see notes on alternatives in next column).	The simple approach suggested in the previous column does not explicitly record negative answers. However, in many cases, a negative response has its own significance and does need to be recorded. - For example, knowing that the patient responded "no" to the question "have you had any pain in the chest?". Similarly, there may be questions that have 3 alternative answers (e.g. "yes", "no", "don't know"). In these cases an approach is needed to distinguish between the available responses. **Note** Options for representing answers such as "yes", "no" and "don't know" include: 1. A postcoordinated expression using the SNOMED CT situation with explicit context model[1] with the symptom represented as the 246090004 |Associated finding| and the answers represented as values of the 408729009 |finding context| attribute (e.g "yes" = 410515003 known present, "no" = 410516002 known absent and "don't know" = 261665006 unknown. 2. A simplified representation of the situation model in which only the finding context values are recorded - e.g. "yes" = 410515003 known present, "no" = 410516002 known absent and "don't know" = 261665006 unknown. 3. A simplified representation of the situation model in which the associated finding is used for "yes" and the finding context is used for "no" and "don't know". For example, "Yes" = 162397003 |sore throat|, "no" = 410516002 known absent and "don't know" = 261665006 unknown.

Terminology Binding

Table 3.2.1-1 highlights the significance of the relationship between the SNOMED CT terminology and EHR structured data and data entry contexts. Formal representations of these relationships are referred to as terminology bindings.

- A terminology binding is a link between a terminology component and an information model artifact.

There are two distinct types of terminology binding:

- A value set binding is a terminology binding that represents the set of permitted values that can be used to populate a coded data item.
  - This type of binding is used to represent search constraints applicable to a particular data entry context.
- A meaning binding is a terminology binding that represents the meaning of a data item or collection of data items.

  Notes
  - SNOMED CT *meaning bindings* associate a meaning, represented by a concept, expression or expression template, with a data item (or a collection of data items) defined by an information model.
  - The meaning represented by an instance of a data item with a *meaning binding* is determined by applying the *meaning binding* to the value in that instance.

- The meaning represented by an instance of a defined collection of data items with a *meaning binding* is determined by applying the *meaning binding* to the values in that instance.

Example

- An application that manages surgical procedure requests could identify requested procedures using concepts that are subtypes of 387713003 |Surgical procedure|. In this case, the data item in which that *concept* is recorded should have a *meaning binding* that explicitly indicates that this is a request (e.g. the expression template below). When this *meaning binding* is applied this data item is interpreted as a subtype of 400999005 |Procedure requested (situation)| and not as a completed procedure.

```
400999005 |Procedure requested (situation)| :
   { 363589002 |Associated procedure (attribute)| = [[ +id < 387713003 |Surgical procedure| ]] }
```

- This type of binding is used to represent aspects of the meaning of the entered data that are derived from the data entry context.

Footnotes

[1] For further information on the situation with explicit context model please refer to Situation with Explicit Context Modeling.

[2] See 3.2.2 EHR Data Entry Design for notes on terminology service requirements for creating data entry templates with bindings to SNOMED CT.

3.2.2 EHR Data Entry Design

This section identifies terminology services required to support the design of data entry forms and templates that incorporate terminology bindings. The objective of these templates is to facilitate the entry of appropriate concepts (or expressions) into electronic health records.

As noted in the previous section, there are a variety of different EHR data entry contexts and each context logically constrains the range of concepts that can be used. This logical constraint can be formally specified using a value set binding represented as an expression constraint. Similarly, the data entry context may affect the meaning of data entered and this can be formally specified using a meaning binding represented by an expression or expression constraint.

A well designed template can simplify data entry and improve data quality by limiting the range of concepts available to those appropriate to a particular data entry context. Table 3.2.2-1 provides examples of some practical uses of templates with terminology bindings.

Data entry templates can also be used to combine and structure related data items, or to enable addition of refinements a selected concept. Different requirements can be met by a common template structure with different terminology bindings. In other cases, a more complex template may be required containing a collection of data items with different value set bindings. Similarly, a meaning binding may be applied to an individual template or to a collection of templates that share a similar interpretation (e.g. past medical history entries). Terminology bound templates that are designed to meet the needs of different data entry contexts can be combined into a more complex template that supports a complete data entry scenario.

Table 3.2.2-1: Practical Uses of Terminology Binding in a Data Entry Template

Practical Use	Examples
To specify a concept or expression to that is added to the record entry when a particular data entry option is selected	• To specify that the concept 195967001 |asthma| should be added to record entry when an item in a check list of medical conditions is marked as true.
To constrain the range of concepts that can be entered into a specific data entry field	• To limit the range of concepts that can be entered in "surgical procedure" field to: ▪ Subtypes of 387713003 |Surgical procedure|; or ▪ Members of a locally defined reference set containing surgical procedures carried out in particular department. • To specify the concepts to be displayed in a short drop-down list from which an item may be selected.
To support the structured entry of specific refinements relevant to a selected concept	• To specify a template that includes questions about general findings followed by prompts for the site of the finding and/or other features. ▪ The template also specifies how the refinement should be represented. Options include: ◦ A record containing several named data items { symptom: 10601006 |Pain| location: 85151006 |left hand|} ; or ◦ An expression such as: 10601006 |Pain| :{ 363698007 |Finding site| = 85151006 |left hand| }
To support the entry of any refinement permitted by the concept model (or a constrained version of the concept model)	• To specify a template that allows a clinical finding to be refined using any attributes that are valid in the clinical finding concept model. ▪ The refined concept would be represented as an expression.

Table 3.2.2-2 shows a summary of the terminology services required to enable terminology bindings to be created and edited when developing user interface templates for health record data entry.

Table 3.2.2-2: Services Required for EHR User Interface Template Editing

Practical Requirement	Status [1]	Required Terminology Services [2]	Additional Terminology Service Dependencies [3]
Select the SNOMED CT edition and version to be used.	REQUIRED	4.1 Select Edition and Version	N/A
Create or edit a user interface template or control that records specific concepts in EHR depending on user selections	REQUIRED	4.8 Find Concepts	N/A
Create or edit a user interface template including expression constraints that limit permitted values that can be entered through a specific data entry control	REQUIRED	4.7 Validate and Apply Expression Constraints	N/A
Create or edit a user interface template or control that records a specific expression in an EHR depending on user selections (in cases where no specific concept matches the required meaning)	OPTIONAL	4.14 Validate Concept Definitions and Expressions	N/A
Get attributes that can be applied to an identified concept	OPTIONAL	4.13 Get Concept Model Rules	N/A
Get the range of values that can be applied to an identified attribute	OPTIONAL	4.13 Get Concept Model Rules	N/A

Footnotes

[1] Applications designed to address this use case must support the practical requirements marked as *Required*. Support for the practical requirements marked as *Optional* is recommended as these provide enhanced functionality that may be required by some users.

[2] In most cases, a reference to a subsection of 4 Terminology Service Types, implies a requirement for all services marked as *Required* in that subsection. However, where a reference is followed by a bulleted list, that list specifies the specific terminology services required. Some of the specific services listed as required for an *Optional* practical requirement may be marked as *Recommended* in the referenced subsection.

[3] The *Additional Terminology Service Dependencies* column contains references to services on which a *Required Terminology Service* depends. This column does not restate dependencies on services listed as required service or additional dependencies for essential requirements listed in earlier rows. A full list of the dependencies of each terminology service is provided in the relevant subsection of 4 Terminology Service Types.

3.2.3 EHR Data Entry

This section identifies the terminology services required to support practical entry of EHR data containing SNOMED CT concepts and expressions. It assumes that templates with appropriate terminology bindings have been created and applied using data entry design techniques identified in 3.2.2 EHR Data Entry Design.

Table 3.2.3-1 shows a summary of the terminology services required to support the most common types of EHR Data Entry. These service requirements are also sufficient to entry of postcoordinated expressions provided that these *expressions* are created using a specific predefined *template*.

Table 3.2.3-1: Terminology Services Required for EHR Data Entry

Practical Requirement	Status [1]	Required Terminology Services [2]	Additional Terminology Service Dependencies [3]
Select the SNOMED CT edition and version to be used for data entry.	REQUIRED	3.1.1 Select Edition and Version	N/A
Enable a concept, and where appropriate an associated term, to be recorded in response to the selection of a particular user interface option.	REQUIRED	4.2 Get a Concept, Description or Relationship • Get concept. 4.3 Get Terms for a Concept	N/A
Display appropriate terms in search results, user interface lists, checkboxes, radio buttons and any other user interface controls in which options are represented by concept identifiers.	REQUIRED	4.3 Get Terms for a Concept	N/A

Enable term searches to be constrained by value set bindings specified for each data entry field.	REQUIRED	4.8 Find Concepts	One or more of the following services may be required depending on the range of search constraint types supported 3.1.5 Get and Test Concept Subtypes and Supertypes • Test subsumption between two concepts; and/or • Test a set of concepts for subsumption 3.1.6 Test Reference Set Membership 3.1.7 Apply Expression Constraints
Enable term searches to be constrained by simple constraints specified by the user to narrow a search.	OPTIONAL		
Enable searches for attribute refinements to be automatically constrained by the appropriate concept model range constraint.	OPTIONAL	3.1.4 Get Definition of a Concept 3.1.8 Find Concepts 4.13 Get Concept Model Rules • Get the range of values applicable to a specified attribute	
Enable the display of the definition of a selected concept.	OPTIONAL	4.4 Get Definition of a Concept	N/A
Enable the display of supertype parents and subtype children of selected items in a search result list.	OPTIONAL	4.5 Get and Test Concept Subtypes and Supertypes • Get supertype parents of a concept • Get subtype children of a concept	N/A
Enable the validation of a generated postcoordinated expression prior to adding it to a record entry.	OPTIONAL	4.14 Validate Concept Definitions and Expressions • Validate expression	N/A

Data entry methods that generate adhoc postcoordinated expressions, without a specific predefined template, have additional requirements for access to the concept definitions and the concept model rules. For example, a user interface control could be designed to respond to the selection of a SNOMED CT concept by displaying options that allow a user to select one or more attributes applicable to the concept's domain. The user could then be prompted for values for each selected attribute allowing them to refine the meaning of the selected concept. Another technique involves natural language processing (NLP) identifying the primary focus concept and any relevant refinements stated in a passage of text[4]. Additional terminology service requirements for these techniques are summarized in Table 3.2.3-2.

Table 3.2.3-2: Additional Terminology Services Required for Adhoc Postcoordinated Data Entry

Practical Requirement	**Status** (for adhoc postcoordination)	**Required Terminology Services**	**Additional Terminology Service Dependencies**

Enable access to the definition of a selected concept.	REQUIRED	4.4 Get Definition of a Concept	N/A
Enable the identification of attributes that can be applied to a selected concept.	REQUIRED	4.13 Get Concept Model Rules • Get the set of attribute rules applicable to an identified concept	N/A
Enable the identification of the range of values that can be applied to a selected attribute.	REQUIRED	4.4 Get Definition of a Concept 4.13 Get Concept Model Rules • Get the range of values applicable to a specified attribute	N/A
Enable the validation of a generated postcoordinated expression prior to adding it to a record entry.	OPTIONAL	4.14 Validate Concept Definitions and Expressions • Validate expression	N/A

Footnotes

[1] Applications designed to address this use case must support the practical requirements marked as *Required*. Support for the practical requirements marked as *Optional* is recommended as these provide enhanced functionality that may be required by some users.

[2] In most cases, a reference to a subsection of 4 Terminology Service Types, implies a requirement for all services marked as *Required* in that subsection. However, where a reference is followed by a bulleted list, that list specifies the specific terminology services required. Some of the specific services listed as required for an *Optional* practical requirement may be marked as *Recommended* in the referenced subsection.

[3] The *Additional Terminology Service Dependencies* column contains references to services on which a *Required Terminology Service* depends. This column does not restate dependencies on services listed as required service or additional dependencies for essential requirements listed in earlier rows. A full list of the dependencies of each terminology service is provided in the relevant subsection of 4 Terminology Service Types.

[4] Natural language processing services are distinct from terminology services are not documented in this guide.

3.3 Display EHR Data

Data stored in an electronic health record (EHR) that includes SNOMED CT concepts or expressions may also include include a human readable term associated with each concept identifier. In this case, EHR data can be displayed without using any terminology services. However, as shown in Table 3.3-1 there are several situations in which it may be necessary to lookup and display a term rather than relying upon a term stored in the record[1].

Table 3.3-1: Reasons for Looking Up Terms when Displaying EHR Data

Reason	Notes

To display the preferred term or fully specified name of a concept in the following situations:	
- If a record entry contains a concept identifier without an associated term	It is good practice to store the selected term as well as the concept identifier but this is not mandatory.
- If a communication contains a concept identifier without an associated term	It is good practice to communicate the selected term as well as the concept identifier but this is not mandatory. In some cases, the design of a message or interchange format may not support the inclusion of terms for all concept identifiers. This may be a particular issue when postcoordinated expressions are communicated as some interchange formats may not support the inclusion of terms for each concept identifier in an expression.
- If the synonym in a record entry or communication is ambiguous	The term stored or communicated may be appropriate to the concept but may not be sufficiently clear. - For example, the term "fundus" is a valid synonym for four different concepts. Looking up the fully specified name for the concept identifier allows the display of term that removes this ambiguity.
To display the preferred term for a concept in a familiar language: - If the term stored or communicated is in an unfamiliar language	The term stored or communicated may be in a language that the user does not understand. If the concept is also included in the edition used by the viewer, it should be possible to display the term in the viewers own language.
To confirm that a term linked to a concept identifier is a valid: - If there is a requirement for quality assurance validation	Even when a term is stored or communicated with a concept identifier, it may be useful to confirm that the term included is associated with that concept identifier[2].

Table 3.3-2 shows a summary of the services required to support display of EHR data including looked up terms.

Table 3.3-2: Services Required to Enable Display of EHR Data

Practical Requirement	Status [3]	Required Terminology Services[4]	Additional Terminology Service Dependencies[5]
Enable the selection of SNOMED CT edition and version to be used	REQUIRED	4.1 Select Edition and Version	N/A
Display terms for the concepts in EHR record entries and communications	REQUIRED	4.3 Get Terms for a Concept	N/A

Footnotes

[1] Term lookup will fail if the concept is a module that is not included in the viewers edition.

[2] Term validation will fail if the term is in a description in a modulethat is not included in the viewers edition (for example the term may be part of a module including term in a language not used by the viewer.

[3] Applications designed to address this use case must support the practical requirements marked as *Required*. Support for the practical requirements marked as *Optional* is recommended as these provide enhanced functionality that may be required by some users.

[4] In most cases, a reference to a subsection of 4 Terminology Service Types, implies a requirement for all services marked as *Required* in that subsection. However, where a reference is followed by a bulleted list, that list specifies the specific terminology services required. Some of the specific services listed as required for an *Optional* practical requirement may be marked as *Recommended* in the referenced subsection.

[5] The *Additional Terminology Service Dependencies* column contains references to services on which a *Required Terminology Service* depends. This column does not restate dependencies on services listed as required service or additional dependencies for essential requirements listed in earlier rows. A full list of the dependencies of each terminology service is provided in the relevant subsection of 4 Terminology Service Types.

3.4 EHR Reporting and Analytics

The design of SNOMED CT facilitates the use of a range of powerful reporting and analytic techniques. For more information about the applications of these techniques, please see the guide to Data Analytics with SNOMED CT. The same techniques can also be employed to support clinical decision support as described in the guide to Decision Support with SNOMED CT.

Table 3.4-1 shows a summary of the terminology services required to support reporting and analytics.

Table 3.4-1: Services Required to Support EHR Reporting and Analytics

Practical Requirement	**Status** [1]	**Required Terminology Services** [2]	**Additional Terminology Service Dependencies** [3]
Enable the selection of SNOMED CT edition and version to be used.	**REQUIRED**	4.1 Select Edition and Version	N/A
Get terms for concepts to use in human readable representations of: • Query specifications • Reports and analytics results	**REQUIRED**	4.3 Get Terms for a Concept	N/A
Apply concept subsumption tests as part of a reporting or analytics query	**REQUIRED**	4.5 Get and Test Concept Subtypes and Supertypes	N/A
Apply reference set membership criteria as part of a reporting or analytics query	**REQUIRED**	4.6 Get and Test Reference Set Membership	N/A
Validate expression constraints	**REQUIRED**	4.7 Validate and Apply Expression Constraints • Validate an expression constraint	N/A

Apply an expression constraint query as part of a reporting or analytics query[4]	REQUIRED	4.7 Validate and Apply Expression Constraints • Get all concepts that conform to an expression constraint	N/A
Find concepts to include in a reporting or analytics query including: • Concept identifiers for use in expression constraints and queries including: ▪ Reference set identifiers for reference set membership tests ▪ Supertypes for subsumption tests ▪ Attributes and value ranges for using in expression constraints and queries	REQUIRED	4.8 Find Concepts • Get concepts using constrained term searches 4.13 Get Concept Model Rules • Create queries that are consistent with concept model rules	N/A
Include subsumed or equivalent expressions in the results of a reporting or analytics query	OPTIONAL	4.15 Test Expression Subsumption	N/A

Footnotes

[1] Applications designed to address this use case must support the practical requirements marked as *Required*. Support for the practical requirements marked as *Optional* is recommended as these provide enhanced functionality that may be required by some users.

[2] In most cases, a reference to a subsection of 4 Terminology Service Types, implies a requirement for all services marked as *Required* in that subsection. However, where a reference is followed by a bulleted list, that list specifies the specific terminology services required. Some of the specific services listed as required for an *Optional* practical requirement may be marked as *Recommended* in the referenced subsection.

[3] The *Additional Terminology Service Dependencies* column contains references to services on which a *Required Terminology Service* depends. This column does not restate dependencies on services listed as required service or additional dependencies for essential requirements listed in earlier rows. A full list of the dependencies of each terminology service is provided in the relevant subsection of 4 Terminology Service Types.

[4] Implementation of expression constraints in a reporting or analytics queries differs from use of a constraints when searching the terminology. When a constraint is applied to a set of instance records it may be necessary to take account of the ways in which data is represented in the EHR. For example, as noted in 3.2 Support EHR Data Entry, the data entry context may have a meaning binding that modifies the interpretation of a particular concept.

3.5 Reference Set Editing

A reference set is a standard format for maintaining and distributing a set of references to SNOMED CT components.

The reference set mechanism provides a standard extensible mechanism that can be used to customize SNOMED CT to meet a wide range of practical use cases[1]. Detailed information about reference sets is provided in the Reference Sets Practical Guide and in section 5.2 Reference Set Types of the SNOMED CT Release File Specifications.

Table 3.5-1 shows a summary of the terminology services required to support reference set creation and editing.

Table 3.5-1: Services Required to Support Reference Set Creation and Editing

Practical Requirement	**Status** [2]	**Required Terminology Services** [3]	**Additional Terminology Service Dependencies** [4]
Enable the selection of SNOMED CT edition and version to be used.	REQUIRED	4.1 Select Edition and Version	N/A
Get concepts or descriptions that are members of a reference set • In the case of a description, this includes the human-readable term for this refset member	REQUIRED	4.2 Get a Concept, Description or Relationship • Get a description	N/A
Get terms for concepts to use in human-readable descriptions of: • The current reference set • Concepts that are members of a reference set	REQUIRED	4.3 Get Terms for a Concept	N/A
Apply concept subsumption tests to identify candidate concepts for addition to or removal from a reference set.	REQUIRED	4.5 Get and Test Concept Subtypes and Supertypes	N/A
Apply reference set membership criteria to identify: • Current members of the reference set being edited • Candidate concepts for addition to or removal from a reference set	REQUIRED	4.6 Get and Test Reference Set Membership	N/A
Create and apply an expression constraint query: • as an intensional definition of reference set membership; or • to identify candidate concepts for addition to or removal from a reference set.[5]	REQUIRED	4.7 Validate and Apply Expression Constraints • Get all concepts that conform to an expression constraint	N/A
Find concepts: • to find the id of a named reference set to be edited • to be added to or removed from a reference set; or • to be included in intensional definition of a reference set	REQUIRED	4.8 Find Concepts	N/A

Get reference set member data (i.e. data of reference sets that extend on the simple type reference set, e.g. data from map type reference sets, association type reference sets etc.)	OPTIONAL	4.10 Get Data from a Reference Set	N/A
Update reference set member data	OPTIONAL	*Refset type specific functionality* [5]	N/A

Footnotes

[1] Requirements and use cases supported by *reference sets* are described in the *Reference Sets Practical Guide* section 3. Requirements and Use Cases. Terminology services that directly support these use cases are described in section 3.1.6, and sections 3.1.10 to 3.1.13. Most other terminology services either directly or indirectly depend on access to reference sets. For example 4.3 Get Terms for a Concept depends on access to members of a language reference set.

[2] Applications designed to address this use case must support the practical requirements marked as *Required*. Support for the practical requirements marked as *Optional* is recommended as these provide enhanced functionality that may be required by some users.

[3] In most cases, a reference to a subsection of 4 Terminology Service Types, implies a requirement for all services marked as *Required* in that subsection. However, where a reference is followed by a bulleted list, that list specifies the specific terminology services required. Some of the specific services listed as required for an *Optional* practical requirement may be marked as *Recommended* in the referenced subsection.

[4] The *Additional Terminology Service Dependencies* column contains references to services on which a *Required Terminology Service* depends. This column does not restate dependencies on services listed as required service or additional dependencies for essential requirements listed in earlier rows. A full list of the dependencies of each terminology service is provided in the relevant subsection of 4 Terminology Service Types.

[5] Terminology services that provide a user interface that enables editing of reference set member data are outside the scope of the current guide. SNOMED International provides a Reference Set Service that is accessible by the SNOMED International community. This enables creation and editing of a Simple Reference Set. Other reference set types include additional data designed to meet a specific purpose. Therefore, editors designed to facilitate creation and maintenance of this additional data are required. One example of the is the SNOMED International Mapping Tool.

3.6 Apply Mapping Reference Sets

This section is considers the terminology services required to apply a mapping reference set to map data between SNOMED CT and another code system.

A mapping reference set is a reference set designed to support the process of mapping data from one code system, classification, or terminology to another code system, classification, or terminology.

- There are several types of *mapping reference set* each of these supports different types of mapping.
    - **Error rendering macro 'sp-plaintextbody-link'**

      Conversion context did not contain original content entity.

    support one-to-one maps to or from SNOMED CT.

- **Error rendering macro 'sp-plaintextbody-link'**

  Conversion context did not contain original content entity.

  support maps where additional information is required to determine the correct mapping from SNOMED CT to a statistical classification such as ICD-10.
- **Error rendering macro 'sp-plaintextbody-link'**

  Conversion context did not contain original content entity.

  support maps to or from another code system where there is an additional requirement to record cases in which a code in one code system was added specifically for the purpose of enabling a map from the codes system in which the code originated.
- **Error rendering macro 'sp-plaintextbody-link'**

  Conversion context did not contain original content entity.

  support maps from other code systems in cases where some or all the source codes may need to be represented by a postcoordinated expression.

Table 3.6-1 shows a summary of the services required to support effective use of a mapping reference set.

In practice, these terminology services only get the relevant mapping data for the source concept (or source code in the other code system). The relevant mapping data may be one or more rows from the mapping reference set. If there are no rows, this implies that there are no maps from the source code. Otherwise, the rows of returned mapping data must be parsed using the rules applicable to the particular map to determine the result. These rules are found in the mapping guide associated with the particular map or mapping reference set type. For example, the mapping rules for the SNOMED International ICD-10 map are documented in the ICD-10 Mapping Technical Guide.

✓ SNOMED International provides an online Mapping Tool that can be used to access maps that it distributes. This tool enables guests to view the maps (see https://mapping.ihtsdotools.org/).

Table 3.6-1: Services Required to Access Mapping Data

Practical Requirement	Status [1]	Required Terminology Services [2]	Additional Terminology Service Dependencies [3]
Enable the selection of SNOMED CT edition and version to be used.	REQUIRED	4.1 Select Edition and Version	N/A
Get maps for a specified concept in an identified map reference set [4]	REQUIRED	4.12 Get Mapping Data • Get maps based on combinations of refsetId and referencedComponentId	3.1.10 Get Data from a Reference Set

Get maps for a specified code in another code system in an identified map reference set[4]	**REQUIRED**	4.12 Get Mapping Data - Get maps based on combinations of refsetId and mapTarget	N/A

When applying a mapping reference set it is important to note that most mapping reference sets are designed to be unidirectional and will not produce reliable results when used to map codes in the opposite direction.

Footnotes

[1] Applications designed to address this use case must support the practical requirements marked as *Required*. Support for the practical requirements marked as *Optional* is recommended as these provide enhanced functionality that may be required by some users.

[2] In most cases, a reference to a subsection of 4 Terminology Service Types, implies a requirement for all services marked as *Required* in that subsection. However, where a reference is followed by a bulleted list, that list specifies the specific terminology services required. Some of the specific services listed as required for an *Optional* practical requirement may be marked as *Recommended* in the referenced subsection.

[3] The *Additional Terminology Service Dependencies* column contains references to services on which a *Required Terminology Service* depends. This column does not restate dependencies on services listed as required service or additional dependencies for essential requirements listed in earlier rows. A full list of the dependencies of each terminology service is provided in the relevant subsection of 4 Terminology Service Types.

[4] Most mapping reference sets are designed to be unidirectional and will not produce reliable results when used to map codes in the opposite direction. Check the documentation before using a mapping reference set and only use it for mapping in the direction specified. For example, the ICD-10 extended map distributed by SNOMED International is designed to be used to map from SNOMED CT to ICD-10.

3.7 Terminology Change Management

The following sections present terminology services use cases related to the management of terminology changes:

3.7.1 Access Details of Terminology Changes

SNOMED CT has rich versioning mechanism that retains the full history of changes to every component and reference set member. As a result, it is possible to review the content of the terminology as it was at any time in the past and to make comparisons between two versions. In addition to tracking the state of the terminology at specific times in the past, the versioning mechanism also provides and indication of the reason for inactivation of each concept or description. In the case of concepts, there is also data linking inactive concepts to active concepts that may be used to replace them. Table 3.7.1-1 shows the terminology services required to access each of these different types of versioning data.

Table 3.7.1-1: Services Required to Access Details of Terminology Changes

Practical Requirement	**Status** [1]	**Required Terminology Services** [2]	**Additional Terminology Service Dependencies** [3]
Enable the selection of SNOMED CT edition and the versions of that edition to be compared	REQUIRED	4.1 Select Edition and Version	N/A
Identify concepts that have been added, changed or inactivated between the specified versions.	REQUIRED	4.9 Identify Changes to the Terminology • Concept	4.2 Get a Concept, Description or Relationship • Get concept by identifier
Get inactivation reason for each inactivated concept	REQUIRED	4.11 Get History Data • Concept inactivation reference set	4.10 Get Data from a Reference Set
Identify concepts that are candidates to replace each inactivated concept	REQUIRED	4.11 Get History Data • Historical association reference sets	N/A
Identify descriptions that have been inactivated between the specified versions.	REQUIRED	4.9 Identify Changes to the Terminology • Description	4.2 Get a Concept, Description or Relationship • Get description by identifier
Get inactivation reason for each inactive description	REQUIRED	4.11 Get History Data • Description inactivation reference set	N/A
Get changes to inferred definitions	OPTIONAL	3.1.9 Identify Changes to the Terminology • Relationship 4.4 Get Definition of a Concept • Get inferred necessary normal form definition of a concept	4.2 Get a Concept, Description or Relationship • Get relationship by identifier
Get changes to stated definitions	OPTIONAL	3.1.9 Identify Changes to the Terminology • OWL reference set 4.4 Get Definition of a Concept • Get stated definition of a concept	4.10 Get Data from a Reference Set • OWL reference set

Footnotes

[1] Applications designed to address this use case must support the practical requirements marked as *Required*. Support for the practical requirements marked as *Optional* is recommended as these provide enhanced functionality that may be required by some users.

[2] In most cases, a reference to a subsection of 4 Terminology Service Types, implies a requirement for all services marked as *Required* in that subsection. However, where a reference is followed by a bulleted list, that list specifies the specific terminology services required. Some of the specific services listed as required for an *Optional* practical requirement may be marked as *Recommended* in the referenced subsection.

[3] The *Additional Terminology Service Dependencies* column contains references to services on which a *Required Terminology Service* depends. This column does not restate dependencies on services listed as required service or additional dependencies for essential requirements listed in earlier rows. A full list of the dependencies of each terminology service is provided in the relevant subsection of 4 Terminology Service Types.

3.7.2 Integrate and Interpret Versioning Data

The services described in 3.7.1 Access Details of Terminology Changes get versioning data related to the addition, modification, or inactivation of individual terminology components. Many practical uses of this data require interpretation of the overall effect of a combination of changes to a concept. Therefore, it may be useful to integrate versioning data in ways that facilitate a review of the impact of these changes. A possible way to achieve this is illustrated by the data model in Figure 3.7.2-1.

A data structure of this type could be used to assist the identification of changes relevant to managing the impact of changes on EHR applications and Extensions in two ways:

- By generating a human-readable version of the change data for use in manual review of existing data
- As a computer processable resource from which queries can be generated to search records, user interface templates, reporting, and analytics queries and extensions for references for changed or inactivated concepts and descriptions.

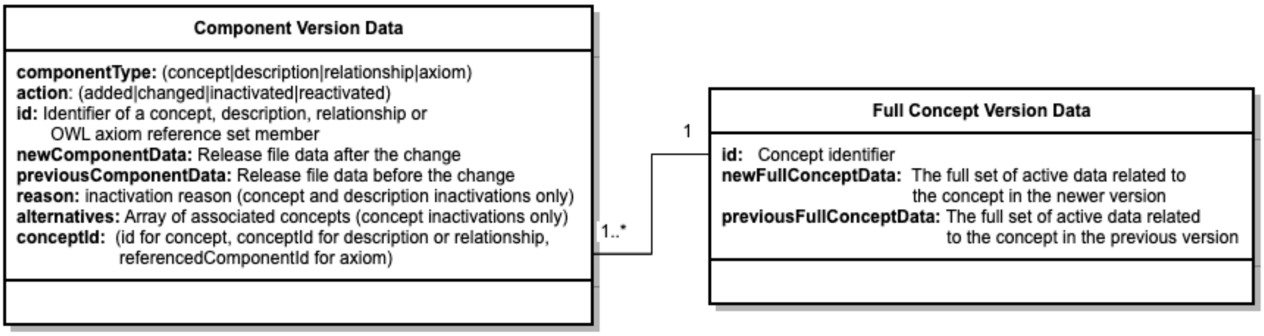

Component Version Data
componentType: (concept|description|relationship|axiom)
action: (added|changed|inactivated|reactivated)
id: Identifier of a concept, description, relationship or OWL axiom reference set member
newComponentData: Release file data after the change
previousComponentData: Release file data before the change
reason: inactivation reason (concept and description inactivations only)
alternatives: Array of associated concepts (concept inactivations only)
conceptId: (id for concept, conceptId for description or relationship, referencedComponentId for axiom)
1..*
1
Full Concept Version Data
id: Concept identifier
newFullConceptData: The full set of active data related to the concept in the newer version
previousFullConceptData: The full set of active data related to the concept in the previous version

Figure 3.7.2-1: Example Data Model for Integrated Versioning Data

Component Version Data

Each instance of the *Component Version Data* object represents the previous and new states of an identified concept, description[1], relationship or OWL axiom reference set member that has been added, changed, inactivated or reactivated between two specified versions of the same edition. Details of the data items in the example *Component Version Data* object are shown in Table 3.7.2-1.

Table 3.7.2-1: Component Version Data Object

Data Item	Notes
componentType	Indicates whether the component is a concept,description, relationship (part of the inferred view of the concept definition), or OWL axiom (part of the stated view of the *concept definition*).
action	Indicates the nature of the change: • added : This component was not present in previous release • changed : The *active* value is unchanged and change has been made to a data value[2] • inactivated : The *active* value of the component has been changed from 1 to 0 • reactivated : The *active* value of the component has been changed from 0 to 1
id	The identifier of the *concept*, *description*, *relationship* or *OWL axiom reference set member*.

newComponentData	The *ComponentData* for the identified *component* in the snapshot release of the later of the two *versions* being compared. **ComponentData** *ComponentData* refers to a representation of the data in the relevant release file row for the identified *concept*, *description*, *relationship* or *OWL axiom*. • In the case of a *description*, the *ComponentData* should also include the acceptability values for that *description* in the language reference sets of that snapshot release. • The data should be represented in a way that supports the data structure of the specified **componentType**. For example, the data could be represented as the string serialization of a JSON object.
previousComponentData	Represents the release file data for the identified *component* in the earlier of the two *versions* being compared. • Empty if the *component* was not present in the earlier *version* of the release file.
reason	Only applicable to *concepts* and *descriptions* that are *active* in the **previousComponentData** view and *inactive* in the **newComponentData** view: • In the case of a *concept,* the reason for inactivation as represented in the *concept inactivation reference set*. • In the case of a *description,* the reason for inactivation as represented in the *description inactivation reference set*.
alternatives	Only applicable to *concepts* that *inactive* in the **newComponentData** view: • An array of historically associated concepts derived by selecting *active* rows from most recent *snapshot view* the *historical association reference sets* with a *referenceComponentId* that matches the identifier of the *concept*. • Each element of the array should include the *refsetId,* which indicates the type of association, and the *targetComponentId,* which refers to associated concept.
conceptId	The identifier of the concept with which this component is associated. This provides a link to the **id** of the relevant **Complete Concept Version Data**. • In the case of a *concept*, this is the same as the *id* (i.e. same as the **id** data item above). • In the case of a *description*, this is the *conceptId*. • In the case of a *relationship,* this is the *sourceId*. • In the case of an *OWL axiom*, this is the *referencedComponentId*.

Full Concept Version Data

There is an instance of the *Full Concept Version Data* object for each concept to which one or more of the following conditions applies:

- The *concept* has been added, changed, inactivated or reactivated
- *OWL axioms* or *relationships* that contribute to the stated and/or inferred *definition* of the *concept* have been added, changed, inactivated or reactivated
- *Descriptions* associated with the *concept* have been added, changed, inactivated or reactivated
- *Language reference set rows* that specify the acceptability of an associated *description* have been added, changed, inactivated or reactivated.

Each instance of the *Full Concept Version Data* object represents the previous and new state of a concept including all its active descriptions (with relevant language acceptability data), all its active relationships and all associated active OWL axioms. Details of the data items in this object are shown in Table 3.7.2-2.

Table 3.7.2-2: Full Concept Version Data Object

Data Item	Notes
id	The identifier of a *concept* that has been affected directly or indirectly by updates between two specified *versions* of the same *edition*.
newFullConceptData	A representation of the *FullConceptData* for the identified *concept* in the snapshot release of the later of the two *versions* being compared. **FullConceptData** *FullConceptData* refers to a representation of the data in the all the relevant active release file rows that describe and define an identified *concept*. - This includes all the data in: - the row of the *concept snapshot release file* with the relevant *id* value. - all active rows of the *description snapshot release file* with a *conceptId* value matching the identifier of the *concept* and for each of these *descriptions* - and all active rows of the *language reference set snapshot release file* with the a *referencedComponentId* value matching the identifier of that *description* - all active rows of the *relationship snapshot release file* with a *sourceId* value matching the identifier of the *concept*. - all active rows of the *OWL reference set snapshot release file* with *refsetId* 733073007 |OWL axiom reference set| and a *referencedComponentId* value matching the identifier of the *concept*. - The data should be represented in a way that supports the data structures of all of the above data components. For example, the data could be represented as the string serialization of a JSON object combining representations of each of the different data items.
previousFullConceptData	A representation of the *FullConceptData* for the identified *concept* in the *snapshot release* of the earlier of the two *versions* being compared. - Empty if this *concept* was not present in the earlier release.

Footnotes

[1] To ensure that changes to the acceptability of a *description* in any language or dialect are captured, a *description* should also be included if any language reference set member that refers to it has been added, changed, inactivated or reactivated.

[2] If the *active* value has changed, the action is either *inactivated* or *reactivated*, even if a data value has also changed.

3.7.3 Manage Impact of Changes on EHR Applications

Table 3.7.3-1 summarizes change management issues that may affect and EHR application when moving to a new version of a SNOMED CT edition. It also outlines approaches to checking for and resolving issues of different types.

Table 3.7.3-1: Impact of Version Updates on EHR Applications

Change Type	Significance	Risks	Factors Affecting Risk Level and Likely Impact	Change Management Actions
Addition of a concept	MODERATE	The new concept may not be available in some data entry contexts. Although all new concepts will be available for ad-hoc searches, it may not be possible to enter a new concept in data entry fields with tightly defined value set bindings.	Risk is higher if the value set binding specifies members of a reference set or individual concepts. Value set bindings that use hierarchy-based constraints are more likely to include new concepts that are appropriate to the data entry context.	Review newly added concepts paying particular attention to concepts in areas of the hierarchies that are frequently used. Consider if new concepts need be added to any reference sets, search constraints, user interface terminology bindings or queries. ✓ **Recommendation** Mitigate the risk of excluding new concepts by using hierarchy constraints, rather than subsets or lists of individual concepts. These will automatically include new concepts that conform to the constraint minimizing the need for manual intervention.
		Inconsistent reporting and analytics may occur. This is a risk if a new concept has not been included in a reporting query criteria.	Risk is higher if reporting queries enumerate members of a reference set or specify each concept individually. Queries that use hierarchy-based constraints are more likely to include new concepts that are relevant to the report.	
Inactivation of a concept	HIGH	Inactive concepts may be explicitly referenced by user interface terminology bindings. This can lead to one of the following errors: • The inactive concept being included in a new records (this is a serious error); or • The system correctly prevents addition of the inactive concept and it becomes impossible to record a significant item of data.	Risk is higher if the value set binding specifies members of a reference set or individual concepts. Value set bindings that use hierarchy-based constraints will automatically exclude inactive concepts.	Check whether any inactivated concepts are used in user interface terminology bindings. Use the reason for inactivation to inform decisions on an appropriate approach to resolution. Update these bindings to remove references to inactive concepts, preferably replacing them with appropriate active concepts identified using historical associations from the replaced concept. For medicolegal reasons these changes should only affect newly entered record.
		Inactive concepts may be referenced in query criteria used for reporting and analytics. This may cause significant changes to the results of those queries, as concepts previously subsumed by (or defined by reference to) this concept may no longer be included.	Risk is higher if the reference to the concept is part of a subsumption constraint. Individual references to the inactive concept will not result in changes to the report output.	Check query criteria used for reporting and analytics for references to inactive concepts. If necessary, adjust the query criteria to delivery intended results (e.g. by excluding inactive concepts).

		Inactive concepts may be present in EHR record entries that were created when the concept was active. The existence of an inactive concept in earlier record entries is **not** an error. However, as the concept is no longer active it will not be subsumed by other concepts and will be excluded from the results of queries that formerly included it.	Use of concepts that are later inactivated. This is a particular issue if the concept is inactivate due to ambiguity because in this case there is no single equivalent active concept.	Review inactivated concepts to determine whether they were previously subsumed by queries used for reporting or analytics. Use the reason for inactivation to inform decisions on an appropriate approach to resolution. Choose an appropriate approach to resolve any potential reporting anomaly. Some possible resolutions are suggested here: • Explicitly include the inactive concept in queries that formerly included it so historical data is still reported consistently. • Create a reference set containing inactive concepts that were previously included in the result of a query and where appropriate include members of that reference set in query criteria. • Use historical associations to extend normal subsumption so that inactive concepts are included whenever their active equivalent concepts are included. In the case of associations in the 900000000000523009 |POSSIBLY EQUIVALENT TO association reference set| this approach may need to be more carefully reviewed and selectively applied. • Accept the exclusion of the now inactive concept from reports. • Map the intended meaning of the original (now inactive) concept to an active concept and add that to the record in a way that allows it to be appropriately included in reports. For medicolegal reasons this should be an addition to and not a replacement of the record.
Change to a concept	NONE	The only mutable concept data attribute in the release file is definitionStatusId. If the value changes this indicates a change in the definition. However, in itself a change to the definitionStatusId is not significant[1].	-	-

Addition of a description	LOW	Makes a new display term available for a concept. This may simply be another option. However, other changes, such as inactivation of other descriptions or changes in language acceptability, may make it necessary to use the new description.	The impact of addition of a new description is minimal unless other descriptions are inactivated or are no longer marked as preferred or acceptable in the relevant language reference set.	Description added to a new concept: • By default the description used to display a new concept should be the preferred term in the selected language or dialect. • Alternative descriptions may be displayed if they are more appropriate to the context or selected by a user. Description added to an existing concept: • Should have no impact on existing stored data. • May be considered as an alternative display term in data entry template or reports.
Change to a description	LOW	Changes to the term and/or caseSignificantId are possible. However, quality assurance rules should limit the range of possible changes to the term in a description.	Changes to descriptions used as display terms in data entry templates or reports.	Review data entry templates and reports that use a changed description as a display term: • Update the display term to match any change to the term. • Update the term if the use of upper and lower case does not conform to the options permitted by the caseSignificanceId. In general, terms stored in existing patient records should not be updated based on terminology changes[2].
Inactivation of a description	HIGH	An inactivated description cannot be used as the source of a display term for data entry or reporting.		Review any use of terms that are associated with: • Descriptions that are inactive • Descriptions that are no longer acceptable or preferred in a selected language reference set. In these cases, replace the use of this term in data entry interfaces with a term for the same concept that is either preferred or acceptable in the selected language reference set. Similarly, if a term is stored with the selected concept, ensure that the term stored is also preferred or acceptable in the selected language. In general, terms stored in existing patient records should not be updated based on terminology changes[2].
Change to the acceptability of a description in a language refset used by the application	MODERATE	A description that is not acceptable for use in one or more of the languages or dialects used in the EHR system, must not be used as the source of a display term for data entry or reporting.		

Additions, changes and inactivations of relationships	MODERATE	Relationships represent the inferred necessary normal form definition of a concept. OWL axiom reference set members represent the stated definitions of concepts. Additions, changes or inactivations of either of these components, represent changes to the definition of a concept. A change in the definition of a concept also affects other concepts with a definitions that refer to that concept. The impact of changes to concept definitions depends on the way these changes affect the results of expression constraints and queries used in data entry templates, reporting and analytics.		Review value set bindings, expression constraints and queries used in reports or analytics and identify the expression constraints on which these depend. For each of these constraints that may have been affected by these changes apply the steps shown in Table 3.7.3-2. Where appropriate fake the actions outlined in Table 3.7.3-3 to address any unintended or unwanted effects arising from the update.
Additions, changes and inactivations of OWL expression refset members				
Additions, changes or inactivations of reference set members	MODERATE	Expression constraints and queries may include or exclude members of a reference set. Additions of new members or inactivation of existing members can therefore alter the results of expression constraints and queries used in data entry templates, reporting and analytics.		

Table 3.7.3-2: Comparing Results of Applying an Expression Constraint to Two Versions of a SNOMED CT Edition

Step	Input	Output Description	Output Name
Identify snapshot views of two versions fo the same SNOMED CT edition	Current edition	Edition in use before planned update	V1-snapshot
	New edition	Edition to be used after planned update	V2-snapshot
Apply the Expression Constraint (ECL) to each version to identify two subsets of concepts that conform to the constraint.	V1-snapshot	Set of concepts in V1-snapshot that conform to the ECL	V1-ECL-result
	V2-snapshot	Set of concepts in V2-snapshot that conform to the ECL	V2-ECL-result
Compare the subsets created by the previous step to identify two subsets that containing concepts that conform to the constraint in one of the versions but do not conform in the other version.	V1-ECL-result	Concepts in V1-ECL-result but not in V2-ECL-result	V1-ECL-only
	V2-ECL-result	Concepts in V2-ECL-result but not in V1-ECL-result	V2-ECL-only
Subdivide the subset of concepts that only conform to the constraint in the current version into two subsets depending on the active status of the concept in the new version.	V1-ECL-only	Concepts in V1-ECL-only that are active in V2-snapshot	V1-ECL-only-V2-active
	V1-ECL-only	Concepts in V1-ECL-only that are inactive in V2-snapshot	V1-ECL-only-V2-inactive
Subdivide the subset of concepts that only conform to the constraint in the new version into three subsets depending on whether the concept is present in the current version and if so depending on the active status of the concept.	V2-ECL-only	Concepts in V2-ECL-only that are active in V1-snapshot	V2-ECL-only-V1-active
	V2-ECL-only	Concepts in V2-ECL-only that are inactive in V1-snapshot	V2-ECL-only-V1-inactive

	V2-ECL-only	Concepts in V2-ECL-only that are absent from V1-snapshot	V2-ECL-only-V1-absent
Check the content of the five sets of concepts generated by the last two steps using the notes in Table 3.7.3-3			

Table 3.7.3-3: Evaluation of Differences between ECL Results

Output Name	Description	Impact	Recommended Action
V2-ECL-only-V1-absent	New concepts in V2 release that were not present in V1. • Inclusion of these concepts will not affect retrospective reports or analytics because the new concepts could not have been used previously[3]. • Inclusion of these new concepts is appropriate as they conform to the stated constraint.	NONE	None
V2-ECL-only-V1-inactive	It is unusual for concepts to be reactivated so this set will usually be empty. Concepts reactivated in V2 release after being inactive in V1 release. However, in theory this could alter the result of retrospective reports because reactivation of the concept causes these concepts to be included whereas previously they were excluded.	LOW	If this subset contains any concepts, be aware that this may affect result of rerunning an earlier reports or analytics on retrospective data. However, in most cases no action is required.
V2-ECL-only-V1-active	Concepts that only conform to the ECL in V2 although they are active in both versions. Retrospective reports will now include concepts that were previously excluded from the report.	HIGH	Concepts in this subset will be included in reports or analytics performed after upgrading to the new version but would not have been included when using the current version. This may be due to an revision of concept definitions or addition of concepts to a reference set. Assuming these changes were intentional improvements the revised result should be more accurate and more complete. In some cases, this type of change may have unintended consequences. Therefore a review of the context in which these constraints are used is recommended. If necessary the constraint can then be refined to exclude some or all of the added concepts.

V1-ECL-only-V2-active	Concepts that only conform to the ECL in V1 although they are active in both versions. Retrospective reports will now exclude concepts that were included in the report.	HIGH	Concepts in this subset will be excluded from reports or analytics performed after upgrading to the new version but would have been included when using the current version. This may be due to an revision of concept definitions or removal of concepts from a reference set. Assuming these changes were intentional improvements the revised result should be more accurate. In some cases, this type of change may have unintended consequences. Therefore a review of the context in which these constraints are used is recommended. If necessary the constraint can then be refined to specifically include some or all of the excluded concepts.
V1-ECL-only-V2-inactive	Concepts that only conform to the ECL in V1 that are inactive in V2. In this case, inactivation of the concept means that retrospective reports will exclude these concepts that were previously included in the report.	MODERATE	Concepts in this subset will be excluded from reports or analytics performed after upgrading to the new version but would have been included when using the current version. In this case, the reason for exclusion is that the concept is inactive rather than being a direct consequence of the constraint. There are two ways to enable an inactive concept to be included to conform to a constraint. A direct reference to a concept identifier or a reference to the members or a reference set that includes that concept[4].

Footnotes

[1] The definitionStatusId fields has two possible values: 900000000000073002 |Sufficiently defined by necessary conditions definition status| or 900000000000074008 |Not sufficiently defined by necessary conditions definition status|. Since July 2019 the information provided by this value is primarily informative and has little practical impact on processing. It indicates whether the necessary conditions defined in the relationships file provide a sufficient definition of the concept. However, the primary representation of the concept definition now uses OWL axioms. These are used to classify the terminology and generate the relationships that represent the inferred definition and to set the definitionStatusId. The change to the OWL axiom is significant and the change to the definitionStatusId is a consequence of that change.

[2] Concept identifiers and terms stored in patient records represent the data as entered and subsequent terminology updates should not result in changes to those original records. This does not preclude additions that reflect the effect of terminology updates on a specific record. However, these additions should be distinct from the original record entries. National and local policies and approaches to this issue may vary depending on legal constraints, clinical guidelines and audit capabilities of EHR systems.

[3] It is possible that a record containing a concept from the new version (V2) might have been received from another EHR instance that was upgraded to V2 earlier. In this case, the receiving system would not include this concept in reports it was upgraded to V2.

[4] In future it may also be possible to include inactive concepts by using the SNOMED Query Language. This query language is being developed to extend the capabilities of expression constraints and is may enable inclusion of inactive concepts that have historical associations with specific active concepts.

3.7.4 Manage Impact of Changes on Extensions

Table 3.7.4-1 summarizes change management issues that may affect a SNOMED CT extension when an edition on which their extension depends is updated. It also outlines approaches to checking for and resolving issues arising from particular types of changes.

> ⚠ The changes outlined in this section must be applied when an extension module is updated to align with the new versions of the modules on which it depends. Before an updated extension module is released, updates must also be made to its module dependencies. Please refer to the Extensions Practical Guide for more detailed information about *extension modules* and the 4.3.2.4.2 Module Dependency Reference Set.

Table 3.7.4-1: Impact of Version Updates on Extensions

Change Type	Significance	Examples of Potential Impact	Change Management Actions
Addition of a concept	MODERATE	A new concept in a module on which the extension depends may have the same meaning as a concept already in the extension.	Consider inactivating the extension concept and creating a historical association from this to the concept concept in the updated module.
		A new concept in a module on which the extension depends may be a supertype of a concept in the extension.	Classification will automatically create new subtype relationships that can be inferred from the stated definition of the extension concept. However, if the definition of an extension concept omits an axiom that is present in the new concept this may not occur. Therefore, new concepts should also be reviewed to identify additional cases whether a new concept should subsume extension concept. Where appropriate, additional axioms should be added to the extension concept to enable accurate classification.
		If the extension includes a language reference set, additions will be required to represent the acceptability of the descriptions associated with the concept.	If necessary add new descriptions to represent this concept in a language or dialect supported by the extension. If the extension includes a language reference set, add references to the fully specified name and preferred term.

Inactivation of a concept	HIGH	A component or reference set member in the extension may refer to a concept that has been inactivated in a module on which the extension depends. • Concept definitions must not refer to inactive concepts. Therefore: ▪ Active OWL expressions must not include references to inactive concepts. ▪ Active relationships must not include references to inactive concepts • Descriptions are permitted to refer to inactive concepts as an inactive concept must retain an active fully-specified name and preferred display term. • Members of some types or reference set member may refer to inactive components. However, references to inactive concepts in the reference sets of the following types should be inactivated and, where appropriate, replaced with members that refer to active concepts: ▪ OWL reference sets. ▪ Mapping reference sets.	OWL reference set rows must be updated to ensure they do not include references to inactive concepts. • Classification will then ensure that no active relationships contain references to inactive concepts.
Addition of a description	NONE	No impact unless the extension includes a language reference sets.	-
	MODERATE	If the extension includes a language reference set this may need to be updated.	If the extension includes a language reference set, consider adding the new description as an acceptable term.
Inactivation of a description	LOW	References to an inactivated descriptions in any reference sets may be affected.	Review and if necessary update reference sets the refer to descriptions that are now inactive.
	MODERATE	If the description is referred to by a language reference set member in the extension module changes may be required to ensure that each concept has an active preferred term and an active fully specified name. If a description that serves one of these roles is inactivated, a new active description must be indicates as its replacement.	If an inactive description is referred to by an active member of a language reference set in the extension, that reference set member should be inactivated. If that language reference set member has the acceptability value "preferred" and refers to an inactive description with type synonym or fully specified name a new active member must be added to refer to an active description of the same type as "preferred".
Changes to concept definitions	NONE	No impact unless the extension includes additional clinical concepts.	-

	HIGH	If the extension includes additional clinical concepts these will need to be reclassified.	If the extension includes additional clinical concepts review the impact of the following on definitions in of concepts in the extension: • Changes changes to the concept model used to define concept in the module. • Changes individual concept definitions on which definitions in the extension depend If necessary update definitions in of concepts in the extension to align with changes in the modules on which it depends. Then reclassify the terminology.

3.8 Support Terminology Authoring and Review

As explained in 2.2.3 Scope of the Guide this guide focuses on read-only services without a user interface. Therefore, this section only outlines the requirements for the more general terminology services that support the authoring process.

It does **not** address services that are specific to authoring such as assignment and management of editing tasks, version management, creating and updating concepts, descriptions and other terminology artifacts. Neither does it address user interface design and related services that enable effective terminology authoring.

Table 3.8-1 shows a summary of the read-only terminology services required to support effective terminology authoring.

Table 3.8-1: Services Required to Support Terminology Authoring and Review

Practical Requirement	Status [1]	Required Terminology Services [2]	Additional Terminology Service Dependencies [3]
Enable selection of SNOMED CT edition and version to be used.	REQUIRED	4.1 Select Edition and Version	N/A
Select supertypes, attributes and values to define a concept	REQUIRED	4.2 Get a Concept, Description or Relationship	N/A
Display terms for concepts in search results, definitions, expressions and expression constraints	REQUIRED	4.3 Get Terms for a Concept	N/A
Get the inferred view as OWL definition following classification	REQUIRED	4.4 Get Definition of a Concept	N/A
Get stated definition of a concept in OWL in authoring environment	REQUIRED	4.4 Get Definition of a Concept	N/A
Get attributes that can be applied to an identified concept	REQUIRED	4.13 Get Concept Model Rules	N/A
Get range of values that can be applied to an identified attribute	REQUIRED	4.13 Get Concept Model Rules	N/A

Footnotes

[1] Applications designed to address this use case must support the practical requirements marked as *Required*. Support for the practical requirements marked as *Optional* is recommended as these provide enhanced functionality that may be required by some users.

[2] In most cases, a reference to a subsection of 4 Terminology Service Types, implies a requirement for all services marked as *Required* in that subsection. However, where a reference is followed by a bulleted list, that list specifies the specific terminology services required. Some of the specific services listed as required for an *Optional* practical requirement may be marked as *Recommended* in the referenced subsection.

[3] The *Additional Terminology Service Dependencies* column contains references to services on which a *Required Terminology Service* depends. This column does not restate dependencies on services listed as required service or additional dependencies for essential requirements listed in earlier rows. A full list of the dependencies of each terminology service is provided in the relevant subsection of 4 Terminology Service Types.

4 Terminology Service Types

This section describes services required to enable effective use of SNOMED CT. The guide does not mandate a particular way to meet these requirements. In some cases, a single terminology service API call may address several of the requirements identified in this section of the guide. In other cases, multiple API calls may be required to address a single requirement.

4.1 Select Edition and Version

Overview

Editions

Several different SNOMED CT editions are available. All editions include the content of the SNOMED CT International Edition but some also include extensions that contain additional content designed for use in a particular country, region, speciality or organization[1]. Terminology services may support access to more than one edition, so the service must enable the client to select the appropriate edition.

It is important to specify the edition to enable access to national or local terminology content that is not available in other editions. Content specific to an edition may include locally added concepts and additional descriptions that provide terms in a national language or local dialect. An edition may also include additional reference sets representing subsets or maps.

Versions

SNOMED CT editions are maintained with regular updates. Each update to an edition is referred to as a SNOMED CT versioned edition. Terminology services should enable access to more than one edition, so the service must enable the client to select the appropriate version.

It is important to specify the version as each version update adds, changes or inactivates some components and reference set members. Therefore, the result of a terminology service may differ depending on the version.

An organization maintaining a SNOMED CT edition or extension will also have internal development versions and prerelease versions for testing (see alpha and beta release package). Therefore, services designed to support terminology development and maintenance also need to enable selection of specific terminology development versions or branches.

Release Types

SNOMED CT specifications define a full release that represents a complete record of the history of all released components, and a snapshot release type that only represents the state of those components at a specified date[2].

Comparing Versions

Some services described in this guide involve checking for changes between versions. Direct support for these services requires two versions to be specified to enable comparison of the before and after state of components that have changed between versions[3].

Requirements and Options

Selection of a specified edition, versioned edition or extended versioned edition is a prerequisite for all other terminology services as it determines the terminology substrate to be used when responding to those service requests. The selection process needs to be supported by services that enable the client application to find out information about available editions, versioned editions and extended versioned editions. The required services are listed in Table 4.1-1.

Table 4.1-1: Services Required

Service Name and Status	Input	Output
Get available editions **REQUIRED**	• None	• Data associated with available SNOMED CT editions: ▪ The name of the edition ▪ An identifier that can be used to select that edition ◦ See note on input to the "Get available versions" service ▪ Optionally other data including: ◦ Information about the organization responsible for maintenance and distribution ◦ Information about available versions of this edition (as specified for the "Get available versions" service)
Get available versions **REQUIRED**	• Edition identifier ⚠ This service must support the use of one or more of the following edition identifiers: • The moduleId of the edition • The URI of the edition (see SNOMED CT URI Standard 2.1 URIs for Editions and Versions) • A edition key returned by the the "Get available editions" service.	• Data associated with available versions of the specified edition: ▪ The name of the edition ▪ The version date ▪ An identifier that can be used to select that *versioned edition* (or *extended versioned edition*): ◦ See note on input to the "Specify a versioned edition" service ▪ Identifiers of the versioned modules included in that version of the edition. ▪ Optionally: ◦ Information about the organization responsible for maintenance and distribution

Service Name and Status	Input	Output
Get available development branches DEVELOPMENT	• Edition identifier	• Data associated with available versions of the specified edition: ▪ The name of the edition ▪ Data related to the development branch ▪ An identifier that can be used to select that development branch using ▪ Optionally: ◦ Information about ownership, status and access rights to the development branch
Specify a versioned edition [4] REQUIRED	• Versioned edition identifier ⚠ This service must support one or more of the following versioned edition identifiers: • The moduleId of the edition and the version date • The URI of the versioned edition (see SNOMED CT URI Standard 2.1 URIs for Editions and Versions) • A key or identifier of the versioned edition returned by the the "Get available versions" service.	In most cases, specification of a versioned edition, extended versioned edition or development branch should be an integral part of a request for another terminology services. It simply identifies the terminology substrate to which those requests apply and does not produce specific output. There are two exceptions to this: • Versioned edition selection services without a request for another service should provide data about the *versioned edition* (or *extended versioned edition*), in a similar as specified for the "Get available versions" service. • If the specified version is not accessible, the output should be an appropriate error message.
Specify an extended versioned edition [4] OPTIONAL	• Versioned edition identifier • moduleId(s) and version dates of each of the modules to be included in the extended versioned edition. ⚠ This creates an informally defined edition consisting of the edition formally specified by the edition moduleId and the contents of one or more compatible extension modules. An extension module is compatible with an edition if: a) Its module dependencies are satisfied by the identified edition; and b) It only include reference sets and metadata concepts. Extension modules that contain clinical terminology content should only be included as part of a formally defined SNOMED CT edition.	
Specify development branch DEVELOPMENT	• Development branch identifier.	

Accessing a Selected Edition Version or Branch

For performance reasons, scalable terminology services should be stateless[5] and this rules out prior configuration of a session to work with a particular edition or version. Therefore, all service requests must explicitly identify the edition and version to be accessed.

- Servers that support access to development versions of an edition must enable access to specified development branches as well as versions.
- Servers that provides access to development branches needs to provide a service that identifies the available development branches.
- Terminology services that support comparison between versions (or between development branches), must allow both the versions (or branches) to be specified.

Interdependencies

Required By

- Other Terminology Services
  - All
- Use Cases
  - All

A terminology server that only provides access to a single snapshot view of pre-specified version and edition of SNOMED CT may not explicitly support this as a service. However, in this case the process of loading data into the service implicitly involves selection of a particular release edition and version.

Depends On

- The terminology server instance from which services are being requested must include data for the versioned editions requested by the client application.

Service Examples

The Snowstorm and FHIR examples are presented in plain text and URL encoded versions. Always use the "Encoded URL" when testing the example service requests. The plain text version is included to aid readability but using this version in a service request may result in errors. These errors result from characters that have to be encoded as they are not permitted in a URL (see IETF RFC1738).

Table 4.1-2: Snowstorm API

Service Name	**API Call** [6]	**Result**
Get available editions	`GET [snowstorm]/codesystems`	Returns JSON representation of all SNOMED CT Editions that are available in the server. The data for each edition includes a *shortname* property used to refer that edition in subsequent API calls. For the International Edition the shortname is "**SNOMEDCT**".

Get available versions of an edition	`GET [snowstorm]/codesystems/[shortname]/versions`
	for example
	`GET [snowstorm]/codesystems/SNOMEDCT/versions`
		Returns JSON representation of each version of the edition specified by *shortname* that are available in the server.
		The data for each edition version includes a *branchPath* property used to refer that edition in subsequent API calls. For the 2020-01-31 release of the International Edition the *branchPath* is "MAIN/2020-01-31". The data returned also includes details of the modules in included in each versioned edition[7].
		Note that URL encoding rules requires the / (slash) character to be replaced with %2F. Thus in subsequent API calls the *branchPath* is represented as **MAIN%2F2020-01-31**

Specify a specific versioned edition	`GET [snowstorm]/codesystems/[branchPath]`
	for example
	`GET [snowstorm]/codesystems/MAIN/2020-01-31`
	Encoded URL
	`GET [snowstorm]/codesystems/MAIN%2F2020-01-31`
		Returns information about the identified versioned edition specified by the *branchPath*. This includes information about all the modules included in the edition.
		In practice the *branchPath* may refer to an extended versioned edition containing additional modules that are not part of the formally defined edition (see 5.2.2 Enabling Access to Extended Editions). In this case, these modules will also be listed.
		ⓘ Other Snowstorm API calls use the same *branchPath* mechanism to refer to specific versioned editions or development branches.

Table 4.1-3: FHIR API

Service Name	API Call[8]	Result
Get available code systems	`GET [fhir]/CodeSystem?_elements=name,url`	Returns a JSON representation of the name and URL of each supported code system. - The URL is then used in other commands to specify the code system to be accessed.

Table 4.1-4: MySQL Example

Service Name	SQL Query[9]	Result
Get available editions	Not supported. Available editions are determined by importing each edition into a separate schema.	-
Specify an edition	To select an edition in the SNOMED CT example database it is necessary to build an instance of the database schema using the set of SNOMED CT release files representing a full release of that edition. Once this schema has been built, queries run on that schema provide access to data in that edition.	-
Specify an extended edition	An edition supplemented by a set of compatible extension modules can be built as a separate schema by importing the full release of the edition and the full release for each of the extension packages required. Once this schema has been built, queries run on that schema provide access to data in that extended edition.	-
Get available versions of an edition	Not required. Any version of that edition, can be accessed from a schema built using the full release files. The version is accessed simply by specifying an appropriate snapshot date.	-

Specify a specific versioned edition	`CALL setSnapshotTime([configId], [snapshotTime]);`
	for example
	`CALL setSnapshotTime(1,'20190731');`
	`CALL setSnapshotTime(2,'20150131');`
		After applying the two procedure calls shown, queries that refer to views with the prefix **snap1_** will be applied to the 2019-07-31 version and queries that refer to views with the prefix **snap2_** will be applied to the 2015-01-31 version.

Any version of the edition supported by the schema, can be accessed provided the schema was built using the full release files. The most recent version and two earlier versions can be accessed simultaneously by referring to different views (snap1 and snap2). The versions in each of the two earlier views are specified in a configuration file. The version settings for each of these the earlier views can be set to specify the version at any past date (see Set Snapshot Time).

Queries run on a specific snapshot view return the data as it was at the configured date.

Note

The SNOMED CT MySQL example database is used as an illustration and is **not** intended for use as a terminology server. In particular, note that the version setting approach described above is **not** stateless. Configuration changes made by any user apply to all the queries they or other users of the database run. This approach to versioning makes it easier to demonstrate processes that require simultaneous access to current and previous versions of an edition. However, this stateful approach is clearly unsafe in a multiuser environment.

Footnotes

[1] SNOMED International Members may maintain National Editions and these may contain additional concepts specific to a particular Member territory and/or descriptions representing international edition concepts in national languages or dialects. Clinical specialty groups, system vendors or healthcare provider organizations editions may maintain extensions that can be combined with the International Edition and/or a National Edition. These extended editions may be limited to additional reference sets or may also include additional content. For more information about SNOMED CT editions and extensions see Appendix B: Editions, Versions and Extensions.

[2] For more information about release types see 3.2 Release Types in the SNOMED CT Release File Specifications.

[3] If a terminology does not support services that compare the state of components in different versions, history related services would require the client application to request the data from both versions and undertake the comparison itself.

[4] A terminology server may support selection of a versioned edition or an edition. In the latter case, the terminology service must also provide services that allow effective time to be specified to ensure the appropriate version of a component is returned.

[5] For a general explanation of the performance and integrity advantages of stateless services see the Wikipedia article on "Service statelessness principle".

[6] In the Snowstorm service requests [snowstorm] should be replaced by the URL to the Snowstorm server endpoint.

[7] In some cases, a Snowstorm path may refer to an extended versioned edition containing additional extension modules that are not part of the formally defined edition (see 5.2.2 Enabling Access to Extended Editions).

[8] In the FHIR service requests [fhir] should be replaced by the URL to the FHIR terminology server endpoint. FHIR® is a registered trademarks of HL7 (www.hl7.org).

[9] For more information about the SNOMED CT example database see the SNOMED CT - SQL Practical Guide. For instructions on how to build the example database refer to Appendix A: Building the SNOMED CT Example Database.

4.2 Get a Concept, Description or Relationship

Overview

The terminology content of SNOMED CT is represented by three types of uniquely identified components:

- A concept is a clinical idea to which a unique concept identifier has been assigned.

- A description is an association between a human-readable phrase (term) and a particular SNOMED CT concept.
- A relationship is an association between a source concept and a destination concept.

  Each of these components is associated with a set of attributes that support interpretation of the component at a given point in time. The data structures of the components and the associations between them are documented in the following section of the SNOMED CT Release File Specifications:

  - 4.2.1 Concept File Specification
  - 4.2.2 Description File Specification
  - 4.2.3 Relationship File Specification
  - 4.1 Associations Between Release Files

Requirements and Options

Getting data associated with identified concepts, descriptions or relationships is a fundamental requirement that must be met by any SNOMED CT terminology services provider. The required services are listed in Table 4.2-1.

Table 4.2-1: Services Required

Service Name and Status	Input	Output
Get concept by identifier **REQUIRED**	• Edition and version • A concept identifier • Optional: Language/dialect[1]	• Concept not found: Return appropriate error message. • Concept found: Return data associated with the concept.
Get description by identifier **REQUIRED**	• Edition and version • A description identifier	• Description not found: Return appropriate error message. • Description found: Return data associated with the description.
Get relationship by identifier **REQUIRED**	• Edition and version • A relationship identifier. • Optional: Language/dialect[1]	• Relationship not found: Return appropriate error message. • Relationship found: Return data associated with the relationship conforming at least to the minimum set specified below: ▪ Minimum: All data for the identified relationship in the snapshot view of the requested edition and version. ▪ Optional additional items: ◦ Preferred term and/or fully specified name for referenced concepts in a specified language ◦ Other data associated with the referenced concepts.

Interdependencies

Required By

- Other terminology services
  - All except 4.1 Select Edition and Version.
- Use cases
  - 3.1 Explore and Review SNOMED CT
  - 3.2.3 EHR Data Entry
  - 3.2.2 EHR Data Entry Design
  - 3.4 EHR Reporting and Analytics
  - 3.5 Reference Set Editing
  - 3.8 Support Terminology Authoring and Review

Depends On

- 4.1 Select Edition and Version

Service Examples

✓ The Snowstorm and FHIR examples are presented in plain text and URL encoded versions. Always use the "Encoded URL" when testing the example service requests. The plain text version is included to aid readability but using this version in a service request may result in errors. These errors result from characters that have to be encoded as they are not permitted in a URL (see IETF RFC1738).

Table 4.2-2: Snowstorm API

Service Name	API Call[2]	Result

Get concept by identifier	`GET [snowstorm]/[branchPath]/concepts/[conceptId]`
	for example
	`GET [snowstorm]/MAIN%2F2020-01-31/concepts/6025007`
	Encoded URL
	`GET [snowstorm]/MAIN%2F2020-01-31/concepts/6025007`
		Returns a JSON representation of data related to the specified concept.
		The data returned:
		- Represents all the current data in the concept file row for the identified concept
		- Also includes the fully specified name and preferred term but does not include other synonyms
		- Includes the definition status but does not include the stated or inferred definition of the concept[3].

Get description by identifier	`GET [snowstorm]/[branchPath]/descriptions/[descriptionId]`
	for example
	`GET [snowstorm]/MAIN%2F2020-01-31/descriptions/2156578010`
	Encoded URL
	`GET [snowstorm]/MAIN%2F2020-01-31/descriptions/2156578010`
		Returns a JSON representation of data related to the specified description.
		The data returned:
		- Represents all the current data in the description file row for the identified description
		- Also includes the language acceptability data for the description derived from each of the language reference sets present in the specified edition
		- Also includes human readable symbolic names for concept enumerations (e.g. " type ": "SYNONYM", "caseSignificance": "CASE_INSENSITIVE")

Get relationship by identifier	`GET [snowstorm]/[branchPath]/concepts/[relationshipId]`
	for example
	`GET [snowstorm]/MAIN%2F2020-01-31/relationships/3303602028`
	Encoded URL
	`GET [snowstorm]/MAIN%2F2020-01-31/relationships/3303602028`
		Returns a JSON representation of data related to the specified relationship.
		The data returned:
		• Represents all the current data in the relationship file row for the identified relationship
		• Also includes the fully specified name and preferred term for the source, type and target concepts

Table 4.2-3: FHIR API

Service Name	API Call[4]	Result

Get SNOMED CT concept by identifier	`GET [fhir]/CodeSystem/$lookup?system=http://snomed.info/sct&code=[conceptId]&_format=json`
	for example
	`GET [fhir]/CodeSystem/$lookup?system=http://snomed.info/sct&code=6025007&_format=json`
		Returns a JSON representation of information about the identified concept
		The data returned includes:
		• The effectiveTime and moduleId of the concept
		• Terms associated with descriptions as a display *designations* of the concept
		• A representation of the subtype relationships of the concept

Table 4.2-4: SQL Illustration

Service Name	SQL Query [5]	Result

Get concept by identifier	`SELECT * FROM snap_concept WHERE id=[conceptId]`
	for example
	`SELECT * FROM snap_concept WHERE id=6025007`
		Returns a row of data from the concept release file for the specified concept.

Get description by identifier	`SELECT * FROM snap_description WHERE id=[descriptionId]`
	for example
	`SELECT * FROM snap_description WHERE id=2156578010`
		Returns a row of data from the description release file for the specified description.

Get relationships by identifier	`SELECT * FROM snap_relationship WHERE id=[relationshipId]`
	for example
	`SELECT * FROM snap_relationship WHERE id=3303602028`
		Returns a row of data from the relationship release file for the specified relationship.

Footnotes

[1] Language and/or dialect should be specified if the service returns terms associated with referenced concepts.

[2] In the Snowstorm service requests [snowstorm] should be replaced by the URL to the Snowstorm server endpoint.

[3] Data in the definitionStatusId and active columns is represented as symbolic names rather than the boolean and SCTID data types used in release file columns.

[4] In the FHIR service requests [fhir] should be replaced by the URL to the FHIR terminology server endpoint. FHIR® is a registered trademarks of HL7 (www.hl7.org).

[5]

The SNOMED CT MySQL example database is **not** designed as a terminology server and is **not** intended for use in a live system . However, it is referenced in this guide as an illustration that some readers may find helpful. For more information about the SNOMED CT example database see the SNOMED CT - SQL Practical Guide. For instructions on how to build the example database refer to Appendix A: Building the SNOMED CT Example Database.

4.3 Get Terms for a Concept

Overview

Each concept is associated with descriptions. Each description is an association between a human-readable phrase (term) and a particular SNOMED CT concept.

There are different types of descriptions and there may be descriptions of each of the following types in several languages:

- A synonym is a word or phrase that expresses the meaning of a SNOMED CT concept in a specified language.
    - Synonyms are terms that represent the meaning of the concept in a way that is familiar and readily understandable. However, when taken out of context a synonym may not be unambiguous and in some cases the same term may be associated with more than one concept.
    - Each concept has at least one synonym and a concepts may have several synonyms. One of the synonym descriptions in each language is marked as the preferred term for the associated concept and others may be marked as acceptable for use. The preferred term is the term deemed to be the most clinically appropriate way of expressing a concept in specified language context.
- A fully specified name is a description that represents the meaning of a concept in a way that is unambiguous and independent of the context in which it is used.
    - Fully specified names are essential for disambiguation of the meaning of a concept. These term used in the fully specified name includes a hierarchy tag that specifies the branch of the SNOMED CT hierarchy the concept is in. The hierarchy tag is enclosed in brackets at the end of the term.
    - There is only one fully specified name in each language. In some languages there may not be a fully specified name, in which case the English fully specified name may be used where necessary for disambiguation.
- A textual definition is a narrative text explanation of the meaning of a concept that may exceed the maximum permitted length for a fully specified name.
    - Textual definitions are only included for some concepts where longer terms are deemed to be necessary or useful to add clarity to the meaning of the concept.

Requirements and Options

Terminology services must enable access to the term in each active description associated with an identified concept. They should also be able to selectively filter the descriptions for which terms are returned based on the following criteria:

- Active status (description.active)
    - Inactive descriptions associated with a concept should not be returned unless explicitly requested.
- Description type (description.typeId)
    - Fully specified name
    - Synonyms (including the preferred term)
    - Textual definitions (and other types where these are used)

- Acceptability in a specified Language Reference Set
  - Preferred: To access the fully specified name or preferred term
  - Acceptable: To access synonyms

The required services are listed in Table 4.3-1.

✓ The services described in 4.2 Get a Concept, Description or Relationship require access to data about individual components based on their unique identifiers. Therefore, a service that gets a description is only required to return the data in the identified description. However, some implementations of those services may also meet some of the requirements for getting terms associated with a concept.

- Services that get concepts may also return some or all of the terms associated with the selected concept.
  - For example, the Snowstorm service that retrieves an identified concept also returns the full y specified name and preferred term for that concept in one language.
  - Similarly, the FHIR service returns a single display term and may optionally also return other associated terms as designations.
- Services that get relationships may also return terms associated with the concepts identified by those relationships.

Table 4.3-1: Services Required

Service Name and Status	Input	Output
Get fully specified name of a concept - Get the fully specified name of a concept in US English (or in a specified language reference set). **REQUIRED** ✓ This requirement is identified as a special case as it is a common requirement. However, it can be met indirectly by either: - *Get concept by identifier* if that service returns the fully specified name in addition to data from the concept file. - *Get filtered terms for a concept* with appropriate filter settings.	- Edition and version - Concept identifier - Language refset set identifier (optional)	- Description not found: Return appropriate error message - Description found: Return data associated with the description: ▪ Required: id, term, caseSignificanceId ▪ Optional: ◦ typeId (which will always be 900000000000207008 | fully specified name|) ◦ languageCode

Service Name and Status	Input	Output
Get preferred term of a concept • Get the preferred term of a concept in US English (or in a specified language reference set). **REQUIRED** ✓ This requirement is identified as a special case as it is a common requirement. However, it can be met indirectly by either: • *Get concept by identifier* if that service returns the preferred term in addition to data from the concept file. • *Get filtered terms for a concept* with appropriate filter settings.	• Edition and version • Concept identifier • Language refset set identifier (optional)	• Description not found: Return appropriate error message • Description found: Return data associated with the description: ▪ Required: id, term, caseSignificanceId Optional: ◦ typeId (which will always be 900000000000013009 | synonym|) ◦ languageCode

Service Name and Status	Input	Output
Get all terms for a concept - Get the terms from all descriptions associated with a concept **REQUIRED**	- Edition and version - Concept identifier - Include inactive descriptions (optional)	- Descriptions not found: Return appropriate error message - Descriptions found: Return data associated with each of the descriptions found ▪ This data must include the id, term, active status, description type, language code and case sensitivity ▪ It should also include the acceptability of the term in one or more language reference sets

Service Name and Status	Input	Output
Get filtered terms for a concept • Get terms from descriptions associated with a particular concept filtered by one or more of the following ▪ Status of the description (*active:* 1=true, 0=false) ▪ Term (*term:* matching a phrase or pattern) ▪ Description type (*typeId:* fully specified name, synonym, textual definition) ▪ Language (*languageCode*) ▪ Acceptability in a specified language reference set (*refset.acceptabilityId:* preferred, acceptable) CONDITIONAL	• Edition and version • Concept identifier • Filters ▪ Active ▪ TypeId ▪ Language code ▪ Language refset identifier ▪ Language refset acceptability	• Descriptions not found: Return appropriate error message • Descriptions found: Return data associated with the descriptions found ▪ This data must include the id, term and case sensitivity ▪ It must also include the term, active status, description type, language code

Service Name and Status	Input	Output
✓ If data necessary to support filtering is provided by "get all descriptions" service, filtering can be done by the client application. In this case, a specific filtered service may not be required.		

Interdependencies

Required By

- Use Cases
  - 3.2.1 Searching, Exploring or Reviewing
  - 3.2.2 EHR Data Entry
  - 3.2.4 EHR Display
  - 3.2.5 EHR Reporting and Analytics
  - 3.2.10 SNOMED CT Authoring

Depends On

- 4.1 Select Edition and Version

ⓘ

- This service requires access to description acceptability data in a language reference set. This could be regarded as a dependency on 4.10 Get Data from a Reference Set. However, access to language reference set data is an essential part of the process of displaying appropriate terms for a concept. Therefore access to the language reference set is included as an integral part of this service and is not declared as a distinct dependency.

Service Examples

✓ The Snowstorm and FHIR examples are presented in plain text and URL encoded versions. Always use the "Encoded URL" when testing the example service requests. The plain text version is included to aid readability but using this version in a service request may result in errors. These errors result from characters that have to be encoded as they are not permitted in a URL (see IETF RFC1738).

Table 4.3-2: Snowstorm API

Service Name	API Call[1]	Result

Get fully specified name of a concept	`GET [snowstorm]/[branchPath]/concepts/[conceptId]`
	for example
	`GET [snowstorm]/MAIN/2020-01-31/concepts/6025007`
	Encoded URL
	`GET [snowstorm]/MAIN%2F2020-01-31/concepts/6025007`
		Returns a JSON representation of data about the selected concept. This include both the fully specified name and the preferred term.
		ⓘ This is the same Snowstorm API call used in 4.2 Get a Concept, Description or Relationship to get concept by identifier. If the fully specified name and/or preferred term there is no need for a specific service to get these terms.

Get preferred term of a concept		

Get all terms for a concept	`GET [snowstorm]/[branchPath]/concepts/[conceptId]/descriptions`
	for example
	`GET [snowstorm]/MAIN/2020-01-31/concepts/6025007/descriptions`
	Encoded URL
	`GET [snowstorm]/MAIN%2F2020-01-31/concepts/6025007/descriptions`
		Returns a JSON representation of data about all descriptions in the selected edition and version that are associated with the concept specified.
		The descriptions returned include:
		• Active and inactive descriptions
		• All types of descriptions
		• Descriptions in all languages that are present in the selected edition
		The data return for each description includes all data present in the description release file and acceptability data derived from language reference sets. The following items have specific roles in filtering descriptions and presenting the terms they contain:
		• The **active** value (true or false) can be used to exclude inactive descriptions.
		• The **typeId** value can be used to filter descriptions of specific types.
		  ▪ The additional a text token **type** provides a human readable representation.
		• The **lang** value contains the language code and this can be used to filter descriptions.
		• The **acceptabilityMap** contains text tokens for the acceptability of the description in one or more language reference sets.
		• The **caseSignificance** value contains a text token indicate whether all or part of a term is case-sensitive. This can be used to enable the case of a term to be adjusted to fit the context of use except in when this would be incorrect or liable to alter the meaning.

Get filtered terms for a concept	Use the API call above to get all terms for a concept and then filter using the returned data.	

ⓘ Snowstorm supports selection of the language(s) in which terms will be returned. This is specified using HTTP request header Accept-Language. The values used with this include the identifiers of the language reference sets to be applied.

For example

{
"Accept": "application/json",
"Accept-Language": "en-X-900000000000509007,en-X-900000000000508004,en"
}

Table 4.3-3: FHIR API

Service Name	API Call[2]	Result

Get all descriptions for a concept	`GET [fhir]/CodeSystem/$lookup?system=http://snomed.info/sct&code=[conceptId]`
	for example
	`GET [fhir]/CodeSystem/$lookup?system=http://snomed.info/sct&code=6025007`
		Returns a JSON representation of data about the selected concept. This include both the fully specified name and all synonyms of the concept[3].
		This is the same FHIR API call used in 4.2 Get a Concept, Description or Relationship to get concept by identifier. If the fully specified name and/or preferred term there is no need for a specific service to get these terms.

Table 4.3-4: MySQL Example Database

Service Name	SQL Query [4]	Result

Get fully specified name of a concept	`SELECT term,caseSignificanceId FROM snap_fsn WHERE conceptId=[conceptId]`
	for example
	`SELECT term,caseSignificanceId FROM snap_fsn WHERE conceptId=6025007`
		Returns the fully specified name of a specified concept and its caseSignificanceId.

Get preferred term of a concept	`SELECT term,caseSignificanceId FROM snap_pref WHERE conceptId=[conceptId]`
	for example
	`SELECT term,caseSignificanceId FROM snap_pref WHERE conceptId=6025007`
		Returns the preferred term of a specified concept and its caseSignificanceId.

Get all synonyms of a concept	`SELECT term,caseSignificanceId FROM snap_synall WHERE conceptId=[conceptId]`
	for example
	`SELECT term,caseSignificanceId FROM snap_synall WHERE conceptId=6025007`
		Returns all the preferred term of a specified concept and with their caseSignificanceId values.
		The view snap_synall includes both preferred and acceptable synonyms. So it includes the the preferred term. The view snap_syn can be used in the same way and this excludes the preferred term.

Get all terms for a concept	`SELECT * FROM snap_description WHERE conceptId=[conceptId]`
	for example
	`SELECT * FROM snap_description WHERE conceptId=6025007`
		Returns all descriptions of all concepts together with all the data for each of those descriptions in the concept file.

Set language configuration
The queries above for fully specified name, preferred terms and synonyms rely on a language configuration to determine which descriptions are used. By default the US English the setting is applied. However, this can be changed using the setLanguage procedure call shown here.
	`CALL setLanguage([configId],[languageCode]);`
	for example
	`CALL setLanguage(0,'en-GB');`
		After this setting is made all references to views including the prefix **snap_** and the suffixes **_fsn**, **_pref**, **_syn**, **_synall** will use the language reference set for GB English (rather than the default language US English).
		Other languages supported by the the installed Edition can also be specified in a similar way.

Footnotes

[1] In the Snowstorm service requests [snowstorm] should be replaced by the URL to the Snowstorm server endpoint.

[2] In the FHIR service requests [fhir] should be replaced by the URL to the FHIR terminology server endpoint. FHIR® is a registered trademarks of HL7 (www.hl7.org).

[3]

Currently the Snowstorm implementation of the FHIR API lookup for a concept does not return language reference set data on the acceptability of different descriptions.

4

The SNOMED CT MySQL example database is **not** designed as a terminology server and is **not** intended for use in a live system . It is referenced in this guide as an illustration that some readers may find helpful. For more information about the SNOMED CT example database see the SNOMED CT - SQL Practical Guide. For instructions on how to build the example database refer to Appendix A: Building the SNOMED CT Example Database.

4.4 Get Definition of a Concept

Overview

Each concept has a concept definition.

- A concept definition is a set of one or more axioms that partially or sufficiently specify the meaning of a SNOMED CT concept.

Requirements and Options

Table 4.4-1 describes several distinct types of access to concept definitions and the client application functionalities that each of these access types supports.

General purpose terminology services that are able to access subtype relationships relationships can be used to meet a limited set of SNOMED CT use cases. However, SNOMED CT terminology services should also provide access to the inferred view of concept definitions.

Advanced SNOMED CT terminology services require the ability to access and interpret the axioms that represent the stated view of concept definitions. They should also enable description logic classification either as a built in service or via an interface to separate classifier. Services that support classification provide added value for data retrieval and analysis by enabling post-coordinated expressions to be classified and appropriately located in the subtype hierarchy.

The required services listed in Table 4.4-2 omit services that only require access to subtype relationships, as these are described in 4.5 Get and Test Concept Subtypes and Supertypes.

Table 4.4-1: Practical Requirements for Access to Concept Definitions

Terminology Service Access to Definitions	Client Application Functionality Supported
Access to subtype relationships: • From subtype child to supertype parent • From supertype parent to subtype child	**Display subtype hierarchy** • A subtype hierarchy is a classification hierarchy in which each node is connected to its supertypes. • *Subtype classification* allows aggregation of information based on a hierarchy of types.

Access to the transitive closure of subtype relationships: - From subtype descendant to supertype ancestor - From supertype ancestor to subtype descendant	**Testing subsumption between concepts** - A subsumption test is a test to determine if a specified candidate concept or expression is a subtype descendant of another specified concept or expression.
Access to the complete inferred view of concept definitions, including subtype relationships and attribute relationships	**Display of inferred concept definition** - The inferred view is a representation of concept definitions that is logically derived by applying a description logic classifier to the stated view. **Postcoordinated expression creation** - A subsumption test is a test to determine if a specified candidate concept or expression is a subtype descendant of another specified concept or expression. **Expression constraint and query creation** - An expression constraint is a computable rule that is used to define a set of clinical meanings.
Access to the complete stated view of concept definitions, as OWL axioms	**Display of stated concept definition** - The stated view is a representation of concept definitions consisting only of assertions made or revised by SNOMED CT authors. **Terminology authoring** - A SNOMED CT authoring is the process of creating or editing SNOMED CT concepts, concept definitions and descriptions. **Terminology classification** - A Description logic classification is a process that generates a set of logically consistent inferences by applying description logic rules to the stated view of concept definitions. **Expression classification** ▪ Applying description logic to classify an expression to enable appropriate subsumption testing when reporting records that include postcoordinated expressions

Table 4.4-2: Services Required[1]

Service Name and Status	Input	Output
Get inferred necessary normal form definition of a concept [2] REQUIRED	- Edition and version - A concept identifier - Optional: Language/ dialect[3]	- The inferred definition of the specified concept represented in one of the following ways: ▪ A SNOMED CT compositional grammar expression[4] ▪ A data object representing the data in all active rows in the snapshot view of the relationship file in which the sourceId column matches the specified conceptId - In either case, the concept identifiers in the definition may optionally be supplemented the preferred term or fully specified name in a specified language or dialect
Get stated definition of a concept ADVANCED	- Edition and version - A concept identifier - Optional: Language/ dialect[3]	- The stated definition of the concept represented in one of the following ways: ▪ As a set of one or more OWL axioms represented using the OWL Functional Syntax or another specified OWL syntax ▪ A SNOMED CT compositional grammar expression - In either case, the concept identifiers in the definition may optionally be supplemented the preferred term or fully specified name in a specified language or dialect

Interdependencies

Required By

- Other Services
  - 3.1.5 Get and Test Concept Subtypes and Supertypes
  - 3.1.6 Test if a Concept or Description is in a Reference Set
  - 4.7 Validate and Apply Expression Constraints
  - 4.8 Find Concepts
  - 4.14 Validate Concept Definitions and Expressions
  - 4.15 Test Expression Subsumption
- Use Cases
  - 3.1 Explore and Review SNOMED CT
  - 3.8 Support Terminology Authoring and Review

Depends On

- 4.1 Select Edition and Version
- 4.2 Get a Concept, Description or Relationship

Service Examples

✓ The Snowstorm and FHIR examples are presented in plain text and URL encoded versions. Always use the "Encoded URL" when testing the example service requests. The plain text version is included to aid readability but using this version in a service request may result in errors. These errors result from characters that have to be encoded as they are not permitted in a URL (see IETF RFC1738).

Table 4.4-3: Snowstorm API

Service Name	API Call[5]	Result

Get the inferred necessary normal form definition of a concept	`GET [snowstorm]/[branch]/concepts/[conceptId]/normal-form?includeTerms=[true|false]`
	for example
	`GET [snowstorm]/MAIN/2020-01-31/concepts/6025007/normal-form?includeTerms=true`
	Encoded URL
	`GET [snowstorm]/MAIN%2F2020-01-31/concepts/6025007/normal-form?includeTerms=true`
	or without including the terms
	`GET [snowstorm]/MAIN/2020-01-31/concepts/6025007/normal-form`
	Encoded URL
	`GET [snowstorm]/MAIN%2F2020-01-31/concepts/6025007/normal-form`
		Returns a JSON data object containing a compositional grammar expression representing the inferred definition of the concept.
		If the optional includeTerms parameter is set to **true** the expression includes the fully specified names of each concept referenced in the expression. Otherwise only the expression only includes the concept identifiers.

Get the stated definition of a concept	`GET [snowstorm]/MAIN%2F2020-01-31/members?referenceSet=733073007&referencedComponentId=[conceptId]`
	for example
	`GET [snowstorm]/MAIN%2F2020-01-31/members?referenceSet=733073007&referencedComponentId=6025007`
	Encoded URL
	`GET [snowstorm]/MAIN%2F2020-01-31/members?referenceSet=733073007&referencedComponentId=6025007`
	or as example of a concept defined by more than one axiom
	`GET [snowstorm]/MAIN%2F2020-01-31/members?referenceSet=733073007&referencedComponentId=473011001`
	Encoded URL
	`GET [snowstorm]/MAIN%2F2020-01-31/members?referenceSet=733073007&referencedComponentId=473011001`
		This service returns a JSON representation of one or more members of in the OWL expression reference set that together represent the stated definition of the identified concept. Each of these reference set members contains an axiom, represented using OWL Functional Syntax, in the additionalFields.owlExpression property string.

Table 4.4-4: FHIR API

Service Name	API Call[6]	Result

Get the inferred necessary normal form definition of a concept	`[fhir]/CodeSystem/$lookup?system=http://snomed.info/sct&code=[conceptId]&property=normalForm&_format=json`
	for example
	`GET [fhir]/CodeSystem/$lookup?system=http://snomed.info/sct&code=6025007&property=normalForm&_format=json`
		Returns a JSON representation of information containing the definition of the concept.
		The data returned includes the necessary inferred, normal form definition of the concept represented as a compositional grammar expression.

Get the stated definition of a concept	N/A	This service is not supported by the FHIR terminology services API

Table 4.4-5: MySQL Example Database

Service Name	SQL Query [7]	Result

<table>
<tr>
<td><b>Get the inferred necessary normal form definition of a concept</b></td>
<td>

```
-- Get inferred defining relationships
SELECT typeId,destinationId,group
FROM snap_relationship
  WHERE AND active=1 AND
 sourceId=[conceptId]
  AND

characteristicType=900000000000011006;
```

for example

```
-- Get inferred defining relationships
SELECT typeId,destinationId,group
FROM snap_relationship
  WHERE active=1 AND

characteristicType=900000000000011006
  AND sourceId=6025007;
```

or including terms

```
-- Get inferred defining relationships with terms
SELECT
  CONCAT(typeId,"|",typeTerm,"|=",
  destinationId,"|",destinationTerm,"|") 'value',
  relationshipGroup 'group'
FROM snap_rel_def_fsn
  WHERE sourceId=6025007;
```

</td>
<td>Returns the set of rows each containing one defining relationship with its associated relationship group number.</td>
</tr>
</table>

Get the stated definition of a concept	`-- Get stated defining axioms`
	`SELECT owlExpression`
	`FROM snap_refset_owlexpression`
	`  WHERE active=1 AND refsetId=733073007`
	`  AND referencedComponentId=[conceptId]`
	`  ORDER BY owlExpression;`
	for example
	`-- Get stated defining axioms`
	`SELECT owlExpression`
	`FROM snap_refset_owlexpression`
	`  WHERE active=1 AND refsetId=733073007`
	`  AND referencedComponentId=6025007`
	`  ORDER BY owlExpression;`
	or as example of a concept defined by more than one axiom
	`-- Get stated defining axioms`
	`SELECT owlExpression`
	`FROM snap_refset_owlexpression`
	`  WHERE active=1 AND refsetId=733073007`
	`  AND referencedComponentId=473011001`
	`  ORDER BY owlExpression;`
		Returns the set of rows each containing one OWL axiom.
		ⓘ The sort order used in this query ensures that class axioms appear before any GCI axioms [7]

Footnotes

[1] Table 4.4-2 only includes services that access the stated or inferred view of a complete concept definition. Services that access subtypes and supertypes of concepts are described in 4.5 Get and Test Concept Subtypes and Supertypes.

[2]

The necessary formal form refers to the logical form in which inferred concept definitions are represented in the relationships file. This form is unable to represent some of the axioms present in the enhanced description logic supported by OWL expressions in the stated view of some concept definitions.

[3] Language and/or dialect should be specified if the service returns terms associated with referenced concepts.

[4] For details of SNOMED CT compositional grammar expressions see Compositional Grammar - Specification and Guide.

[5] In the Snowstorm service requests [snowstorm] should be replaced by the URL to the Snowstorm server endpoint.

[6] In the FHIR service requests [fhir] should be replaced by the URL to the FHIR terminology server endpoint. FHIR® is a registered trademarks of HL7 (www.hl7.org).

[7] The SNOMED CT MySQL example database is not designed as a terminology server and is not intended for use in a live system . It is referenced in this guide as an illustration that some readers may find helpful. For more information about the SNOMED CT example database see the SNOMED CT - SQL Practical Guide. For instructions on how to build the example database refer to Appendix A: Building the SNOMED CT Example Database.

[8] For more information about the differences between SubClass axioms and GCI axioms see General Concept Inclusions (GCIs) in the Editorial Guide.

4.5 Get and Test Concept Subtypes and Supertypes

Overview

The inferred definition of each concept includes a set of one or more subtype relationships relationships.

A subtype relationship is a relationship that asserts that a concept is a subtype of another concept.

- *Subtype relationships* are represented by relationship type 116680003 |is a|.
- A *subtype relationship* asserts that a concept conforms to all the defining characteristics the supertypeconcept but also has at least one feature or refinement that distinguishes it from that *concept*.
- *Subtype relationships* are transitive. Each individual *subtype relationship* links a concept to a supertype parent concept. That concept will also have one or more subtype relationships to its own supertype parents.
- These transitive chains of subtype relationships link every active concept to a set of supertype ancestors and eventually to the |SNOMED CT root concept|.

The meaning of a concept subsumes the meaning of its subtype children and descendants. Therefore, service that test or traverse subtype relationships are essential for effective meaning-based reporting and analysis.

Requirements and Options

Display of the SNOMED CT hierarchy requires access to the direct supertype parents and subtype children of a specified concept.

Effective use of SNOMED CT for retrieval, analysis and constrained data entry requires rapid identification of concepts that are supertype ancestors or subtype descendants of a specified concept.

The requirements listed in Table 4.5-1 include services that return the full set of children, parents, descendants or ancestors of a concept. However, other requirements can be more efficiently met by testing for a transitive subtype or supertype relationship between a specified concept and a set of one or more candidate concepts.

Table 4.5-1: Services Required[1]

Service Name and Status	Input	Output
Get subtype children of a concept REQUIRED	• conceptId • Optionally language or dialect code[2]	• Set of conceptIds of all concepts with a direct 116680003 |is a| relationship to the specified concept. • Optionally additional information such as the fully specified name or preferred term of each concept in the set.
Get supertype parents of a concept REQUIRED	• conceptId • Optionally language or dialect code[2]	• Set of conceptIds of all concepts with a direct 116680003 |is a| relationship from the specified concept. • Optionally additional information such as the fully specified name or preferred term of each concept in the set.
Get subtype descendants of a concept REQUIRED	• conceptId • Optionally language or dialect code[2]	• Set of conceptIds of all concepts with a direct or transitive 116680003 |is a| relationship to the specified concept. • Optionally additional information such as the fully specified name or preferred term of each concept in the set.
Get supertype ancestors of a concept REQUIRED	• conceptId • Optionally language or dialect code[2]	• Set of conceptIds of all concepts with a direct or transitive 116680003 |is a| relationship from the specified concept. • Optionally additional information such as the fully specified name or preferred term of each concept in the set.
Test subsumption between two concepts RECOMMENDED[3]	• Predicate.conceptId • Candidate.conceptId	• If Candidate concept is a subtype descendant of the predicate concept: TRUE • Otherwise: FALSE
Test a set of concepts for subsumption RECOMMENDED[3]	• Predicate.conceptId • Set of candidate.conceptIds	• Set of conceptIds of all concepts in the candidate set that have a direct or transitive 116680003 |is a| relationship from the specified concept.

Interdependencies

Required By

- Other Services
  - 4.7 Validate and Apply Expression Constraints
  - 4.8 Find Concepts
  - 4.14 Validate Concept Definitions and Expressions
  - 4.15 Test Expression Subsumption
- Use Cases
  - 3.1 Explore and Review SNOMED CT
  - 3.2.3 EHR Data Entry
  - 3.4 EHR Reporting and Analytics

Depends On

  - 4.1 Select Edition and Version
  - 4.2 Get a Concept, Description or Relationship
  - 4.4 Get Definition of a Concept

Service Examples

The Snowstorm and FHIR examples are presented in plain text and URL encoded versions. Always use the "Encoded URL" when testing the example service requests. The plain text version is included to aid readability but using this version in a service request may result in errors. These errors result from characters that have to be encoded as they are not permitted in a URL (see IETF RFC1738).

Table 4.5-2: Snowstorm API

Service Name	**API Call**[4]	**Result**

Get subtype children of a concept	`GET [snowstorm]/browser/[branch]/concepts/[conceptId]/children`
	for example
	`GET [snowstorm]/browser/MAIN/2020-01-31/concepts/80146002/children`
	Encoded URL
	`GET [snowstorm]/browser/MAIN%2F2020-01-31/concepts/80146002/children`
		Returns a JSON representation of data about each of the child concepts.
		The data returned for each concept includes:
		• All concept release file data[5]
		• The preferred term and fully specified name.
		Also returns the **total** number of child concepts.

Get supertype parents of a concept	`GET [snowstorm]/browser/[branch]/concepts/[conceptId]/parents`
	for example
	`GET [snowstorm]/browser/MAIN/2020-01-31/concepts/80146002/parents`
	Encoded URL
	`GET [snowstorm]/browser/MAIN%2F2020-01-31/concepts/80146002/parents`
		Returns a JSON representation of data about each of the parent concepts.
		The data returned for each concept includes:
		• All concept release file data[5]
		• The preferred term and fully specified name.
		Also returns the **total** number of parent concepts.

Get subtype descendants of a concept	`GET [snowstorm]/[branch]/concepts/[conceptId]/descendants`
	for example
	`GET [snowstorm]/MAIN/2020-01-31/concepts/80146002/descendants?limit=50&offset=0`
	Encoded URL
	`GET [snowstorm]/MAIN%2F2020-01-31/concepts/80146002/descendants?limit=50&offset=0`
		Returns a JSON representation of data about each of the descendant concepts.
		The data returned for each concept includes:
		• All concept release file data[5]
		• The preferred term and fully specified name.
		Also returns the **total** number of descendant concepts.
		As some concepts have very large numbers of descendants, this service is paged. Requests parameters include:
		• **limit** to restrict the number of descendants returned (default 50).
		• **offset** to specify the start in the results (in multiples of the limit).

Get supertype ancestors of a concept	`GET [snowstorm]/browser/[branch]/concepts/[conceptId]/ancestors`
	for example
	`GET [snowstorm]/browser/MAIN/2020-01-31/concepts/80146002/ancestors`
	Encoded URL
	`GET [snowstorm]/browser/MAIN%2F2020-01-31/concepts/80146002/ancestors`
		Returns a JSON representation of data about each of the ancestor concepts.
		The data returned for each concept includes:
		• All concept release file data[5]
		• The preferred term and fully specified name.
		Also returns the **total** number of ancestor concepts.

Test subsumption between two concepts	Snowstorm does not provide a specific service for this test but the test can be accomplished using the get concepts service (see 4.7 Validate and Apply Expression Constraints) with a combination of an expression constraint to represent the subsumption requirement and a concept identifiers list.
	`GET [snowstorm]/MAIN/2020-01-31/concepts?ecl=<[predicateSupertypeId]&conceptIds=[candidateSubtypeId]`
	For example
	`GET [snowstorm]/MAIN/2020-01-31/concepts?ecl=<404684003&conceptIds=703264005`
	Encoded URL
	`GET [snowstorm]/MAIN%2F2020-01-31/concepts?ecl=%3C404684003&conceptIds=703264005`
		Returns a JSON object with contents that depend on whether the candidate concept is a subtype of the predicate concept:
		• If it is a subtype descendant the returned object contains:
		▪ The property **total** with value 1; and
		▪ An item array that contains data about the candidate concept (as it is the only concept that is both a subtype of the predicate and equal to the candidate concept).
		• If it is **not** a subtype descendant the returned object contains:
		▪ The property **total** with value 0.

Test a set of concepts for subsumption	Snowstorm does not provide a specific service for this test but the technique used in the row above can be used with all the candidate concept identifiers included in the comma separated list.
	`GET [snowstorm]/MAIN/2020-01-31/concepts?ecl=<[predicateSupertypeId]&conceptIds=[candidateSubtypeIdList]`
	For example
	`GET [snowstorm]/MAIN/2020-01-31/concepts?ecl=<404684003&conceptIds=703264005,307581005,195967001`
	Encoded URL
	`GET [snowstorm]/MAIN%2F2020-01-31/concepts?ecl=%3C404684003&conceptIds=703264005%2C307581005%2C195967001`
		Returns a JSON representation of the candidate concepts that are subtypes descendants of the predicate concept.
		• If one or more of the predicate concepts are subtypes of the candidate concept the returned object contains:
		▪ The property **total** with value equal to the number of candidate concepts that are subtypes of the predicate concept; and
		▪ For each candidate concept that is a subtype of the predicate concept a an item array that contain data about that candidate concept .
		• If none of the candidate subtypes are a subtype descendants the returned object contains:
		▪ The property **total** with value 0.
		As some expression constraints are matched by large numbers of concept, this service is paged. Requests parameters include:
		• **limit** to restrict the number of concepts returned (default 50).
		• **offset** to specify the start in the results (in multiples of the limit).

Table 4.5-3: FHIR API

Service Name	API Call[6]	Result

Get subtype children of a concept	FHIR does not provide a specific operation for this service, but the SNOMED CT Expression Constraint Language supports that retrieval of the direct children of SNOMED CT concept, and can thus be used to enable this service through the ValueSet/$expand operation.
	`GET [fhir]/ValueSet/$expand?url=[editionURI/versionURI]?fhir_vs=ecl/`
	for example:
	`GET [fhir]/ValueSet/$expand?url=http://snomed.info/sct?fhir_vs=ecl/`
		Returns a JSON representation of data about each of the subtype children concepts.
		The data returned for each concept includes:
		• **code:** the code for each subtype child
		• **display:** the preferred term for subtype child
		Also returns the **total** number of children concepts.
		As some concepts have very large numbers of children, this service is paged. Requests parameters include:
		• **count** to restrict the number of children returned.
		• **offset** to specify the start in the results (in multiples of the limit).

Get supertype parents of a concept	FHIR does not provide a specific operation for this service, but the SNOMED CT Expression Constraint Language supports that retrieval of the direct parents of SNOMED CT concept, and can thus be used to enable this service through the ValueSet/$expand operation.
	`GET [fhir]/ValueSet/$expand?url=[editionURI/versionURI]?fhir_vs=ecl/>![predicateSubtypeId]&count=10`
	for example:
	`GET [fhir]/ValueSet/$expand?url=[editionURI/versionURI]?fhir_vs=ecl/>!26322001&count=10`
		Returns a JSON representation of data about each of the parent concepts.
		The data returned for each concept includes:
		• **code:** the code for each parent
		• **display:** the preferred term for each parent
		Also returns the **total** number of parent concepts.
		As some concepts have very large numbers of parents, this service is paged. Requests parameters include:
		• **count** to restrict the number of parents returned.
		• **offset** to specify the start in the results (in multiples of the limit).

Get subtype descendants of a concept	**Option 1)**
	`GET [fhir]/ValueSet/$expand?url=[editionURI/versionURI]?fhir_vs=isa/[predicateSupertypeId]&count=10`
	for example:
	`GET [fhir]/ValueSet/$expand?url=http://snomed.info/sct?fhir_vs=isa/27624003&count=10`
		Returns a JSON representation of data about each of the descendant concepts.
		The data returned for each concept includes:
		• **code:** the code for each descendant
		• **display:** the preferred term for each descendant
		Also returns the **total** number of descendant concepts.
		As some concepts have very large numbers of descendants, this service is paged. Requests parameters include:
		• **count** to restrict the number of descendants returned.
		• **offset** to specify the start in the results (in multiples of the limit).

	Option 2)
	`GET [fhir]/ValueSet/$expand?url=[editionURI/versionURI]?fhir_vs=ecl/[predicateSupertypeId]&count=10`
	`GET [fhir]/ValueSet/$expand?url=[editionURI/versionURI]?fhir_vs=ecl/<26322001&count=10`	

Get supertype ancestors of a concept	**Option 2)**
	FHIR does not provide a specific operation for this service, but the SNOMED CT Expression Constraint Language supports that retrieval of the ancestors of SNOMED CT concept, and can thus be used to enable this service through the ValueSet/$expand operation.
	`GET [fhir]/ValueSet/$expand?url=[editionURI/versionURI]?fhir_vs=ecl/>[predicateSubtypeId]&count=10`
	for example
	for example:
	`GET [fhir]/ValueSet/$expand?url=[editionURI/versionURI]?fhir_vs=ecl/>26322001&count=10`
		Returns a JSON representation of data about each of the ancestor concepts.
		The data returned for each concept includes:
		• **code:** the code for each descendant
		• **display:** the preferred term for each descendant
		Also returns the **total** number of descendant concepts.
		As some concepts have very large numbers of ancestors, this service is paged. Requests parameters include:
		• **count** to restrict the number of descendants returned.
		• **offset** to specify the start in the results (in multiples of the limit).

Test subsumption between two concepts	`GET [fhir]/CodeSystem/$subsumes?system=[systemURI]&version=[versionURI]&codeA=[conceptId]&codeB=[conceptId]`
	for example:
	`GET [fhir]/CodeSystem/$subsumes?system=http://snomed.info/sct&version=http://snomed.info/sct/900000000000207008/version/20200131&codeA=307355007&codeB=118940003`
		Returns a JSON representation of information about the subsumption relation between the two concepts:
		• **subsumed-by** indicates that the concept provided as the value for codeA is a subtype of the concept provided as the value for codeB
		• **subsumes** indicates that the concept provided as the value for codeA is a supertype of the concept provided as the value for codeB
		• **not-subsumed** indicates that there is no subsumption relationship between the concepts provided as values for codeA and codeB

Table 4.5-4: MySQL Example Database

Service Name	**SQL Query** [7]	**Result**

Get subtype children of a concept	`SELECT id,term FROM`
	`snap_rel_child_fsn`
	`WHERE`
	`conceptId=[conceptId]`
	for example
	`SELECT id,term FROM`
	`snap_rel_child_fsn`
	`WHERE conceptId=80146002;`
		Returns rows containing the id and term for each subtype child of the specified concept.
		• The query shown return the fully specified name as the term for each concept
		• Replace the view suffix _fsn with _pref to return the preferred term for each concept

Get supertype parents of a concept	`SELECT id,term FROM`
	`snap_rel_parent_fsn WHERE`
	`conceptId=[conceptId]`
	for example
	`SELECT id,term FROM`
	`snap_rel_parent_fsn WHERE`
	`conceptId=80146002;`
		Returns rows containing the id and term for each supertype parent of the specified concept.
		• The query shown return the fully specified name as the term for each concept
		• Replace the view suffix _fsn with _pref to return the preferred term for each concept

Get subtype descendants of a concept	`SELECT id,term FROM`
	`snap_tc_descendant_pref WHERE`
	`conceptId=[conceptId]`
	for example
	`SELECT id,term FROM`
	`snap_tc_descendant_pref`
	`WHERE conceptId=80146002;`
		Returns rows containing the id and term for each subtype descendant of the specified concept.
		• The query shown return the preferred term for each concept
		• Replace the view suffix _pref with _fsn to return the fully specified name for each concept

Get supertype ancestors of a concept	`SELECT id,term FROM`
	`snap_tc_ancestor_pref`
	`WHERE`
	`conceptId=[conceptId]`
	for example
	`SELECT id,term FROM`
	`snap_tc_ancestor_pref`
	`WHERE conceptId=80146002;`
		Returns rows containing the id and term for each supertype ancestor of the specified concept.
		• The query shown return the preferred term for each concept
		• Replace the view suffix _pref with _fsn to return the fully specified name for each concept

Test subsumption between two concepts	`SELECT count(supertypeId) FROM snap_transclose WHERE supertypeId=[predicateConceptId] AND subtypeId=[candidateConceptId]`
	for example
	`SELECT count(supertypeId) FROM snap_transclose WHERE supertypeId=80146002 AND subtypeId=6025007;`
		Returns 1 if the candidate concept is a subtype of the supertype concept.

Test a set of concept for subsumption	Different approaches can be used depending on the specific requirement. As with the options for 1. Use the get descendants service to generate the list of descendants of the predicate concept and use this list as part of a query for records containing subtypes of that concept; 2. Individually, apply the test for subsumption between two concepts to each candidate concept; 3. Extend the ECL query to include all the candidate concepts as illustrated below. `SELECT subtypeId FROM snap_transclose WHERE supertypeId=80146002 AND subtypeId IN (703264005, 307581005, 195967001,6025007)`	Option 3 will return the ids of concepts in the bracketed list that are subtypes descendants of the predicate concept.

Footnotes

[1] The services in this table that list subtype and supertype concepts are a subset of the services required to implement expression constraints (see see 4.7 Validate and Apply Expression Constraints). Therefore, it is recommended that the Expression Constraint Language syntax is used when requesting these services.

[2] Language and/or dialect should be specified if the service returns terms associated with referenced concepts.

[3] The subsumption test services are marked as *recommended* rather than *required* because they can be implemented by the client application using the results of the get subtype descendants and/or get supertype ancestors services.

[4] In the Snowstorm service requests [snowstorm] should be replaced by the URL to the Snowstorm server endpoint.

[5] Data in the definitionStatusId and active columns is represented as symbolic names rather than the boolean and SCTID data types used in release file columns.

[6] In the FHIR service requests [fhir] should be replaced by the URL to the FHIR terminology server endpoint. FHIR® is a registered trademarks of HL7 (www.hl7.org).

[7] The SNOMED CT MySQL example database is not designed as a terminology server and is not intended for use in a live system . It is referenced in this guide as an illustration that some readers may find helpful. For more information about the SNOMED CT example database see the SNOMED CT - SQL Practical Guide. For instructions on how to build the example database refer to Appendix A: Building the SNOMED CT Example Database.

4.6 Get and Test Reference Set Membership

Overview

Reference set files provide a standard format for maintaining and distributing a set of references to SNOMED CT components.

Requirements and Options

A concept or description is a member of a reference set if that reference set contains an active row with a referencedComponentId matching the id of that concept or description.

Two services are specified in this section. One of these returns the list of members of a specified reference set. The other tests whether a candidate concept or description is a member of a specified reference set.

The union, intersection or complement of two or more reference sets can be determined using the SNOMED CT Expression Constraint Language (ECL). Terminology service requirements for accessing expression constraints are described in 4.7 Validate and Apply Expression Constraints.

The services listed in Table 4.6-1 can be used to determine whether a concept or description is a member of any identified reference set. These services are primarily applicable to simple reference sets that represent subsets of concepts or descriptions (5.2.1 Simple Reference Set). Members of other types of reference set contain additional data and services that provide access to this additional data are described in 4.10 Get Data from a Reference Set .

Table 4.6-1: Services Required

Service Name and Status	Input	Output
Get all members of a specified reference set **REQUIRED**	- A reference set specified by its refsetId	- A list of concept or description ids [1] - Option to include additional information about each concept or description

Service Name and Status	Input	Output
Test if a concept or description is a member of specified reference set **REQUIRED**	- A reference set specified by its refsetId - A candidate concept.id or description.id	- If the candidate concept or description is a member of the reference set: TRUE [2] - Otherwise: FALSE

Interdependencies

Required By

- Other Services
  - 4.7 Validate and Apply Expression Constraints
  - 4.8 Find Concepts
- Use Cases
  - 3.2.3 EHR Data Entry
  - 3.4 EHR Reporting and Analytics
  - 3.5 Reference Set Editing

Depends On

- 4.1 Select Edition and Version

Service Examples

✓ The Snowstorm and FHIR examples are presented in plain text and URL encoded versions. Always use the "Encoded URL" when testing the example service requests. The plain text version is included to aid readability but using this version in a service request may result in errors. These errors result from characters that have to be encoded as they are not permitted in a URL (see IETF RFC1738).

Table 4.6-2: Snowstorm API

Service Name	API Call [3]	Result

Get all members of a specified reference set	`GET [snowstorm]/[branch]/members?active=true&referenceSet=[refsetId]`
	for example
	`GET [snowstorm]/MAIN/2020-01-31/members?active=true&referenceSet=723264001`
	Encoded URL
	`GET [snowstorm]/MAIN%2F2020-01-31/members?active=true&referenceSet=723264001`
		Returns a JSON representation of data about all the active members of the reference set.
		It also returns the **total** number of members in the reference set.
		As some reference sets contain a large numbers of members, this service is paged. Requests parameters include:
		• **limit** to restrict the number of reference set members returned (default 50).
		• **offset** to specify the starting point in the results (in multiples of the limit).
		This example returns all active members of the 723264001 |Lateralizable body structure reference set| . This is large reference set with nearly 20,000 active members so although this call returns the count of members it only returns data on a limited number of members [4] .

Test if a concept or description is a member of specified reference set	`GET [snowstorm]/[branch]/members?active=true&referencedComponentId=[candidateId]&referenceSet=[refsetId]`
	for example this call returns true
	`GET [snowstorm]/MAIN/2020-01-31/members?active=true&referencedComponentId=53120007&referenceSet=723264001`
	Encoded URL
	`GET [snowstorm]/MAIN%2F2020-01-31/members?active=true&referencedComponentId=53120007&referenceSet=723264001`
	while the next call returns false
	`GET [snowstorm]/MAIN/2020-01-31/members?active=true&referencedComponentId=80891009&referenceSet=723264001`
	Encoded URL
	`GET [snowstorm]/MAIN%2F2020-01-31/members?active=true&referencedComponentId=80891009&referenceSet=723264001`
		Returns a JSON representation of data including a count of matching member rows followed by data from the matching member rows. The result of the test can be determined by checking the **total** property of the returned object. Non-zero implies the candidate concept or description is a member of the reference set, zero implies it is not.
		ⓘ The two example calls both test membership of the 723264001 |Lateralizable body structure reference set| . The first one returns with a total value of 1 confirming that 53120007 |Upper limb structure| is a member of this reference set. The second example returns with a total value of 0 as 80891009 |Heart structure| is not a member of this reference set.

Table 4.6-3: FHIR API

Service Name	API Call[5]	Result

Get all members of a specified reference set	`GET [fhir]/ValueSet/$expand?url=http://snomed.info/sct?fhir_vs=refset/[refsetId]&count=10`
	for example
	`GET [fhir]/ValueSet/$expand?url=http%3A%2F%2Fsnomed.info%2Fsct%3Ffhir_vs%3Drefset%2F721144007&count=10`
	An alternative solution is to use the expression constraint language, as shown here:
	`GET [fhir]/ValueSet/$expand?url=http%3A%2F%2Fsnomed.info%2Fsct%3Ffhir_vs%3Decl%2F%5E%5BrefsetId%5D&count=10`
	`GET [fhir]/ValueSet/$expand?url=http%3A%2F%2Fsnomed.info%2Fsct%3Ffhir_vs%3Decl%2F%5E721144007&count=10`
		Returns a JSON representation of data about each of the reference set member.
		The data returned for each concept includes:
		• **code:** the conceptId of the reference set member
		• **display:** the preferred term for the reference set member
		Also returns the **total** number of reference set members
		As some reference sets have very large numbers of children, this service is paged. Requests parameters include:
		• **count** to restrict the number of members returned.
		• **offset** to specify the start in the results (in multiples of the limit).

Test if a concept is a member of specified reference set	FHIR does not provide a specific operation for this service, but the SNOMED CT Expression Constraint Language supports testing for reference set membership, and can thus be used to enable this service through the ValueSet/$expand operation.
	`GET [fhir]/ValueSet/$expand?url=http://snomed.info/sct?fhir_vs=ecl/^[refsetId] AND [predicateConceptId]`
	for example
	`GET [fhir]/ValueSet/$expand?url=http://snomed.info/sct?fhir_vs=ecl/^721144007 AND 734009000`
	Encoded URL
	`GET [fhir]/ValueSet/$expand?url=http%3A%2F%2Fsnomed.info%2Fsct%3Ffhir_vs%3Decl%2F%5E721144007+AND+734009000`
	Note that the ECL memberOf function needs to be url encoded. The URL encoding for ^ is %5E
	If multiple predicate concepts should be tested in the same requests this can be done using the following expression constraint:
	`^[refsetId] AND ([predicateConceptId_1] OR [predicateConceptId_2].... OR [predicateConceptId_n])`
		Returns a JSON representation of data about each of the predicate concepts that are members of the reference set.
		The data returned for each concept includes:
		• **code:** the conceptId of the reference set member
		• **display:** the preferred term for the reference set member
		Also returns the **total** number of concepts satisfying the expression constraint.
		Note: If none of the predicate concepts are which are tested for membership are included in the reference set, the service will return 0 results.

Table 4.6-4: MySQL Example Database

Service Name	SQL Query [6]	Result

Get all members of a specified reference set	`SELECT referencedComponentId`
	`FROM snap_refset_simple`
	`    WHERE active=1 AND`
	` refsetId=[refsetId];`
	for example
	`SELECT referencedComponentId`
	`FROM snap_refset_simple`
	`    WHERE active=1 AND`
	` refsetId=723264001;`
		Returns the ids of all the concepts or descriptions that are the members of the reference set.

Test if a concept or description is a member of specified reference set	`SELECT count(referencedCompone`
	`ntId)`
	`    FROM snap_refset_simple`
	`    WHERE active=1 AND`
	` refsetId=[refsetId]`
	`    AND`
	`referencedComponentId=[candida`
	`teComponentId];`
	for example
	`SELECT count(referencedCompone`
	`ntId)`
	`    FROM snap_refset_simple`
	`    WHERE active=1 AND`
	` refsetId=723264001`
	`    AND`
	`referencedComponentId=53120007`
	`;`
		Returns:
		• 0 : if the candidate component is not in the reference set.
		• 1 : If the candidate component is a member of the reference set
		▪ Some types or reference set can include the same component more than once, so any value greater than zero indicate the component is a member of the references set.

Footnotes

[1] The reference set specification should indicate which component types are permitted to be members. The members of some reference sets are concepts while the members of other reference sets are concepts. It is also possible, for some reference sets to contain both concepts and descriptions as members. Services that only return the referencedComponentId do not need to be aware of the types of component in the reference set. However, if a service returns terms or other additional data, the service must take account of the component type.

[2] The value TRUE may be represented in a variety of ways. For example: as a Boolean value, as a non-zero count of matching members or by a data object representation of the candidate component. Similarly the value FALSE may be represented by a Boolean, by a zero count, or by an empty or null returned object.

[3] In the Snowstorm service requests [snowstorm] should be replaced by the URL to the Snowstorm server endpoint.

[4]

The number of results returned defaults to 50 but can be set using a limit parameter in the call. Additionally the starting offset within a large set of results can be specified by an additional parameter.

[5] In the FHIR service requests [fhir] should be replaced by the URL to the FHIR terminology server endpoint. FHIR ® is a registered trademarks of HL7 (www.hl7.org).

[6] The SNOMED CT MySQL example database is not designed as a terminology server and is not intended for use in a live system . It is referenced in this guide as an illustration that some readers may find helpful. For more information about the SNOMED CT example database see the SNOMED CT - SQL Practical Guide. For instructions on how to build the example database refer to Appendix A: Building the SNOMED CT Example Database.

4.7 Validate and Apply Expression Constraints

Overview

An expression constraint is a computable rule that is used to define a set of clinical meanings.

- *Expression constraints* can be used as:
  - formal constraints on the content of a particular data element in an electronic health record.
  - intensional definitions of concept-based reference sets.
  - machine processable queries that identify a set of matching precoordinated expressions or postcoordinated expressions.
  - constraints that restrict the range of an attribute defined in the SNOMED CT concept model.

The Expression Constraint Language - Specification and Guide defines the syntax used to represent expression constraints. It also defines the rules for applying constraints specified using expression constraint language (ECL) to SNOMED CT concepts and expressions.

Requirements and Options

Effective use of SNOMED CT requires the ability to constrain searches for concepts in a variety of ways. Subtype constraints and reference set membership constraints are identified as specific service requirements in sections 4.5 Get and Test Concept Subtypes and Supertypes and 4.6 Get and Test Reference Set Membership. Expression constraints can represent combinations of those constraints. They can also include rules that apply to specific aspects of the definition of a concept. For example, limiting concepts to those that are associated with particular body structures.

Expression constraints can be applied for a range of different purposes. In some cases, these may require enumeration of the set of all the concepts that conform to the constraint, while in other cases it may be more appropriate to apply a specific test to an individual concept. The required services are listed in Table 4.7-1.

Table 4.7-1: Services Required

Service Name and Status	Input	Output
Validate an expression constraint RECOMMENDED [1]	• Edition and version • Expression constraint	• Error message if expression constraint contains syntax errors • Error message if any concept identifiers in the expression constraint are not present in the specified edition

Service Name and Status	Input	Output
Get all concepts that conform to an expression constraint REQUIRED	- Edition and version - Expression constraint - Optional: Language/dialect[2]	- Set of conceptIds of all concepts that conform to the expression constraint - Optionally additional information such as the fully specified name or preferred term of each concept in the set. Error message if expression constraint contains syntax errors
Test if a specific concept conforms to an expression constraint RECOMMENDED[3]	- Edition and version - Expression constraint - Candidate concept identifier	- A true or false result depending on whether the candidate concept conforms to the constraint

Interdependencies

Required By

- Other Services
  - 4.8 Find Concepts
  - 4.14 Validate Concept Definitions and Expressions
  - 4.15 Test Expression Subsumption
- Use Cases
  - 3.2.2 EHR Data Entry Design
  - 3.4 EHR Reporting and Analytics
  - 3.5 Reference Set Editing

Depends On

- 4.1 Select Edition and Version
- 4.2 Get a Concept, Description or Relationship
- 4.3 Get Terms for a Concept
- 4.4 Get Definition of a Concept
- 4.5 Get and Test Concept Subtypes and Supertypes
- 4.6 Get and Test Reference Set Membership

Service Examples

✓ The Snowstorm and FHIR examples are presented in plain text and URL encoded versions. Always use the "Encoded URL" when testing the example service requests. The plain text version is included to aid readability but using this version in a service request may result in errors. These errors result from characters that have to be encoded as they are not permitted in a URL (see IETF RFC1738).

Table 4.7-2: Snowstorm API

Service Name	API Call[4]	Result

Validate an expression constraint	Not supported as individual service. Limited validation is possible by applying an expression constraint (as shown in next row).	An error message is returned if the expression constraint contains a syntax error. The error message does not indicate the specific cause of the validation error. No error message is shown if a concept identified in the expression constraint is not present in the selective edition or version [7].

Get all concepts that conform to an expression constraint	`GET [snowstorm]/snomed-ct/[branchPath]/concepts?activeFilter=true&ecl=[expressionConstraint]`
	The two examples are shown first as the expression constraint complete with the terms for each of the concepts. This is followed by the API service calls in which the terms have been removed (as an optional simplification). The constraint has been URL encoded as required for the REST API.
	Bone fractures with sites that are members of the lateralizable body structure reference set
	< 125605004 |Fracture of bone| : 363698007 |Finding site| = ^ 723264001 |Lateralizable body structure reference set|
	`GET [snowstorm]/MAIN/2020-01-31/concepts?activeFilter=true&ecl=<125605004:363698007=^723264001`
	Encoded URL
	`GET [snowstorm]/MAIN%2F2020-01-31/concepts?activeFilter=true&ecl=%3C125605004%3A363698007%3D%5E723264001`
	Bone fractures with sites that are NOT members of the lateralizable body structure reference set
	< 125605004 |Fracture of bone| : 363698007 |Finding site| != ^ 723264001 |Lateralizable body structure reference set|
	`GET [snowstorm]/MAIN/2020-01-31/concepts?activeFilter=true&ecl=<125605004:363698007!=^723264001`
		Returns a JSON representation of data related to the concepts that conform to the expression constraint.
		The data returned for each concept includes:
		• All concept release file data[5]
		• The preferred term and fully specified name.
		Also returns the **total** number of concepts that match the constraint.
		As some expression constraints are matched by large numbers of concept, this service is paged. Requests parameters include:
		• **limit** to restrict the number of concepts returned (default 50).
		• **offset** to specify the start in the results (in multiples of the limit).
		With the data from the 2020-01-31 International release
		• The first example returns 1442 concepts.
		• The second example returns 737 concepts.
		If the expression constraint contains a syntax error, the returned data is an error message.

Encoded URL

```
GET [snowstorm]/
MAIN%2F2020-01-31/concepts?
activeFilter=true&ecl=%3C1256
05004%3A363698007%21%3D%5E723
264001
```

ⓘ Note that the ECL memberOf function needs to be url encoded. The URL encoding for ^ is %5E

Test if a specific concept conforms to an expression constraint	The service used for getting concepts that conform to an expression constraint can be adapted to test whether one or more specific concepts conform to the constraint by simply extending the constraint so it also constrains the results to only include the specified concept(s).
	Test if fracture of mandible (jaw bone) is a bone fracture with a site that is a member of the lateralizable body structure reference set
	(< 125605004 |Fracture of bone| : 363698007 |Finding site| = ^ 723264001 |Lateralizable body structure reference set|) and (263172003 |Fracture of mandible|)
	`GET [snowstorm]/MAIN/2020-01-31/concepts?ecl=(<125605004:363698007=^723264001) and (263172003)`
	Encoded URL
	`GET [snowstorm]/MAIN%2F2020-01-31/concepts?ecl=%28%3C125605004%3A+363698007%3D%5E723264001%29+and+%28263172003%29`
	Test if fracture of femur (thigh bone) is a bone fracture with a site that is a member of the lateralizable body structure reference set
	(< 125605004 |Fracture of bone| : 363698007 |Finding site| = ^ 723264001 |Lateralizable body structure reference set|) and (71620000 |Fracture of femur|)
	`GET [snowstorm]/MAIN/2020-01-31/concepts?ecl=(<125605004:363698007=^723264001) and (71620000)`
	Encoded URL
		Returns a JSON representation of data related to the concept if it conforms to the expression constraint as described above.
		When testing a single candidate concept the value of the property **total** is sufficient to tell whether the candidate concept conforms to the constraint:
		1. Fracture of mandible **total:** 0
		▪ It is a subtype fracture of bone but the mandible is not lateralizable
		2. Fracture of femur **total**: 1
		▪ It is a subtype fracture of bone and the femur is lateralizable (there a left femur and a right femur)
		3. Osteomyelitis of femur **total** : 0
		▪ Femur is lateralizable and is the finding site but the osteomyelitis is not a subtype of fracture.

```
GET [snowstorm]/
MAIN%2F2020-01-31/concepts?
ecl=%28%3C125605004%3A+363698
007%3D%5E723264001%29+and+
%2871620000%29
```

Test if osteomyelitis of femur is a bone fracture with a site that is a member of the lateralizable body structure reference set

```
(< 125605004 |Fracture of bone| :
363698007 |Finding site| = ^ 723264001 |
Lateralizable body structure reference set| ) and
( 1551001 |Osteomyelitis of femur| )
```

```
GET [snowstorm]/MAIN/
2020-01-31/concepts?
ecl=(<125605004:
363698007=^723264001) and
(1551001)
```

Encoded URL

```
GET [snowstorm]/
MAIN%2F2020-01-31/concepts?
ecl=%28%3C125605004%3A+363698
007%3D%5E723264001%29+and+
%281551001%29
```

Test which of a specific set of concepts conform to an expression constraint	The technique used for testing whether a single concept conforms to an expression constraint can also be used to determine which members of a specified set of concepts conform to that constraint. In the case each of the member of the set of candidates is included in the constraint as shown below:
	(< 125605004 |Fracture of bone| : 363698007 |Finding site| = ^ 723264001 | Lateralizable body structure reference set|) and (263172003 |Fracture of mandible| or 71620000 |Fracture of femur| or 1551001 |Osteomyelitis of femur| or 37449000 |Open fracture of ulna|)
	`GET [snowstorm]/MAIN/2020-01-31/concepts?ecl=(<125605004:363698007=^723264001) and (263172003 or 71620000 or 15510011551001 or 37449000)`
	Encoded URL
	`GET [snowstorm]/MAIN%2F2020-01-31/concepts?ecl=%28%3C125605004%3A+363698007%3D%5E723264001%29+and+%28263172003+or+71620000+or+15510011551001+or+37449000%29`
		Returns a JSON representation of data related to concepts in the set that conforms to the expression constraint.
		The example shown returns an array containing data about the following concepts:
		• 71620000 |Fracture of femur|
		• 37449000 |Open fracture of ulna|

Table 4.7-3: FHIR API

Service Name	API Call[5]	Result
Validate an expression constraint	N/A	Not supported as individual service. Limited validation is possible by applying an expression constraint (as shown in next row).

Get all concepts that conform to an expression constraint	`GET [fhir]/ValueSet/$expand?url=[versionURI]?fhir_vs=ecl/[expressionConstraint]`
	Example 1
	Bone fractures with sites that are members of the lateralizable body structure reference set
	< 125605004 |Fracture of bone| :
	363698007 |Finding site| =^ 723264001 |Lateralizable body structure reference set|
	`GET [fhir]/ValueSet/$expand?url=http://snomed.info/sct/900000000000207008/version/20200131?fhir_vs=ecl/<125605004:`
	`363698007=^723264001`
	Encoded URL
	`GET [fhir]/ValueSet/$expand?url=http%3A%2F%2Fsnomed.info%2Fsct%2F900000000000207008%2Fversion%2F20200131%3Ffhir_vs%3Decl%2F%3C125605004%3A%0A363698007%3D%5E723264001`
	Example 2
	Bone fractures with sites that are NOT members of the lateralizable body structure reference set
	< 125605004 |Fracture of bone| : 363698007 |Finding site| != ^ 723264001 |Lateralizable body structure reference set|
	`GET [fhir]/ValueSet/$expand?url=http://snomed.info/sct/900000000000207008/version/20200131?fhir_vs=ecl/<125605004:`
	`363698007!=^723264001`
	Encoded URL
	`GET [fhir]/ValueSet/$expand?url=http%3A%2F%2Fsnomed.info%2Fsct%2F900000000000207008%2Fversion%2F20200131%3Ffhir_vs%3Decl%2F%3C125605004%3A%0A363698007%21%3D%5E723264001`
		Returns a JSON representation of data about each of the concepts in the specified version that conform to the expression constraint.
		The data returned for each concept includes:
		- **code:** the conceptId of the reference set member
		- **display:** the preferred term for the reference set member
		Also returns the **total** number of reference set members
		As some expression constraints are satisfied by many concepts, this service is paged. Requests parameters include:
		- **count** to restrict the number of members returned.

			• **offset** to specify the start in the results (in multiples of the limit). • **Example 1** returns 1442 concepts. • **Example 2** returns 737 concepts.

The /ValueSet/$validate-code operation can be used to test whether a concept is included in a specific value set, where the value set is identified as an implicit value set.

```
GET [fhir]/ValueSet/$validate-code?system=http://snomed.info/
sct&code=[conceptId]&url=[versionURI]?fhir_vs=ecl/
[expressionConstraint]
```

Example 3

Test if fracture of mandible (jaw bone), 263172003 |Fracture of mandible|, is a bone fracture with a site that is a member of the lateralizable body structure reference set.

The subset of concepts that are bone fractures with a site that is a member of the lateralizable body structure reference set can be determined by the following expression constraint:

```
< 125605004 |Fracture of bone| :
363698007 |Finding site| = ^ 723264001 |Lateralizable body structure reference set|
```

The resulting FHIR request:

```
GET [fhir]/ValueSet/$validate-code?system=http://snomed.info/
sct&code=1551001&url=http://snomed.info/sct/
900000000000207008/version/20200131?fhir_vs=ecl/<125605004:
363698007=^723264001
```

Encoded URL

```
GET [fhir]/ValueSet/$validate-code?
system=http%3A%2F%2Fsnomed.info%2Fsct&code=1551001&url=http%3A
%2F%2Fsnomed.info%2Fsct%2F900000000000207008%2Fversion%2F20200
131%3Ffhir_vs%3Decl%2F%3C125605004%3A%0A363698007%3D%5E7232640
01
```

Example 4

Test if fracture of femur (thigh bone), 71620000 |Fracture of femur|, is a bone fracture with a site that is a member of the lateralizable body structure reference set

```
GET [fhir]/ValueSet/$validate-code?system=http://snomed.info/
sct&code=71620000&url=http://snomed.info/sct/
900000000000207008/version/20200131?fhir_vs=ecl/<125605004:
363698007=^723264001
```

Encoded URL

- Returns a JSON representation of data about whether the concept is included in the set.
- **result**: true, if the concept is included and false if the concept is not included
- **display:** the display term for the concept
- With the data from the 2020-01-31 International release

- **Example 3**
  - **result:** false
- **Example 4**

	`GET [fhir]/ValueSet/$validate-code?system=http%3A%2F%2Fsnomed.info%2Fsct&code=71620000&url=http%3A%2F%2Fsnomed.info%2Fsct%2F900000000000207008%2Fversion%2F20200131%3Ffhir_vs%3Decl%2F%3C125605004%3A%0A363698007%3D%5E723264001`	▪ **result**: True ▪ **display:** "Fracture of femur"

Test which of a specific set of concepts conform to an expression constraint	The /ValueSet/$validate-code operation only supports testing whether a single concept conforms to an expression constraint. Therefore, to determine which members of a specified set of concepts conform to an expression constraint, a dedication expression constraint needs to be designed to enable this test. In this case, each member of the set of candidates is included in the expression constraint as shown below:
	`([intensionalDefinition])AND([candidate1] OR [candidate2] OR [candidate3] OR ...[candidateN])`
	The following general request template will test which of the candidates matches the expression constraint represented in the [intensionalDefinition]
	`GET [fhir]/ValueSet/$expand?system=http://snomed.info/sct&code=[conceptId]&url=[versionURI]?fhir_vs=ecl/[expressionConstraint]`
	Example 5
	Test which of the following concepts *is a bone fracture with a site that is a member of the lateralizable body structure reference set*
	• 263172003 |Fracture of mandible|
	• 71620000 |Fracture of femur|
	• 1551001 |Osteomyelitis of femur|
	• 37449000 |Open fracture of ulna|
	(< 125605004 |Fracture of bone| : 363698007 |Finding site| = ^ 723264001 |Lateralizable body structure reference set|) and (263172003 |Fracture of mandible| or 71620000 |Fracture of femur| or 1551001 |Osteomyelitis of femur| or 37449000 |Open fracture of ulna|)
	The resulting FHIR request:
	`GET [fhir]/ValueSet/$expand?url=http://snomed.info/sct/900000000000207008/version/20200131?fhir_vs=ecl/(<125605004:363698007=^723264001) and (263172003 or 71620000 or 1551001 or 37449000)`
	Encoded URL
	`GET [fhir]/ValueSet/$expand?url=http%3A%2F%2Fsnomed.info%2Fsct%2F900000000000207008%2Fversion%2F20200131%3Ffhir_vs%3Decl%2F%28%3C125605004%3A%0A363698007%3D%5E723264001%29+and+%28263172003+or+71620000+or+1551001+or+37449000%29`
		Returns a JSON representation of data about each of the candidate concepts that conform to the expression constraint.
		The data returned for each concept includes:
		• **code:** the conceptId of the reference set member
		• **display:** the preferred term for the reference set member
		Also returns the **total** number of concepts satisfying the expression constraint.
		Note: If none of the candidate concepts which are tested for membership are included in the reference set, the service will return 0 results.
		The request in **Example 5** shows that the following two concepts conform to the expression constraint:
		• 71620000 | Fracture of femur|

		- 37449000 |Open fracture of ulna|

Table 4.7-4: MySQL Example Database

Service Name	SQL Query[6]	Result
Validate an expression constraint	Not supported.	

Get all concepts that conform to an expression constraint	`call eclQuery([expressionConstraint])`
	Examples
	Bone fractures with sites that are members of the lateralizable body structure reference set
	`call eclQuery("<125605004|Fracture of bone|: 363698007|Finding site| = ^723264001|Lateralizable body structure reference set|")`
	Bone fractures with sites that are members of the lateralizable body structure reference set
	`call eclQuery("<125605004|Fracture of bone|: 363698007|Finding site| = ^723264001|Lateralizable body structure reference set|")`
		Returns a row of data containing the conceptId and term for each concept that conforms to the constraint.
		If the expression constraint is invalid, no rows are returned.
		ⓘ This and the following service examples can also use the following alternative procedure calls:
		The procedure eclQueryCount() sets a count output parameter with the number of concepts that match the constraint.
		`call eclQueryCount([expressionConstraint],@count)`
		The procedure eclSelect() returns the concepts that match the constraint in a specified style. Styles include: exp (expression with preferred term), expfsn (expression with FSN), pref (id, preferred term), fsn (id,FSN), allsyn (id,synonym - one row per synonym).
		`call eclSelect([expressionConstraint],[style])`
		The procedure eclGetIds() outputs the concept identifiers to rows in the table config_resultsets with keyed by by a specified setId. This allows other queries or procedures to access the results.
		`call eclQueryGetIds([expressionConstraint],[setId])`

Test if a specific concept conforms to an expression constraint	`call eclQuery(([expressionConstraint]) and ([conceptId]))`
	for example
	`call eclQuery("(<125605004=^723264001) AND (71620000)")`
		If the concept conforms to the constraint, this returns a row of data containing that conceptId and its term. If it does not conform no rows are returned.

Test which of a set of specified concepts conform to an expression constraint	`call eclQuery("(<125605004:363698007=^723264001) AND (71620000) or (15510011551001) or (37449000)")`	Returns the conceptId and term for concepts in the set that conform to the expression constraint.

Footnotes

[1] The validation service is recommended rather than required as minimal validation can be provided by attempting to apply the expression constraint. However, a validation service that reports specific errors, is required for use cases that involve creating or editing expression constraints (e.g 3.2.2 EHR Data Entry Design).

[2] Language and/or dialect should be specified if the service returns terms associated with referenced concepts.

[3] The test service is recommended rather than required because this can be delivered by the get service simply by adding an additional constraint as shown in the Snowstorm illustration.

[4] In the Snowstorm service requests [snowstorm] should be replaced by the URL to the Snowstorm server endpoint.

[5] In the FHIR service requests [fhir] should be replaced by the URL to the FHIR terminology server endpoint. FHIR® is a registered trademarks of HL7 (www.hl7.org).

[6] The SNOMED CT MySQL example database is not designed as a terminology server and is not intended for use in a live system. It is referenced in this guide as an illustration that some readers may find helpful. For more information about the SNOMED CT example database see the SNOMED CT - SQL Practical Guide. For instructions on how to build the example database refer to Appendix A: Building the SNOMED CT Example Database.

[7] Snowstorm expression validation has been raised as an open issue see https://github.com/IHTSDO/snowstorm/issues/145#issue-676721969.

Bone fractures with sites that are NOT members of the lateralizable body structure reference set

4.8 Find Concepts

Overview

Finding a specific concept is a fundamental requirement and a prerequisite for most other terminology service. To make practical use of SNOMED CT, users must be able search for concepts using using phrases, words or parts of words. The SNOMED CT Search and Data Entry Guide provides detailed descriptions of these essential search types. It also notes a range of important additional features that facilitate appropriate use of the terminology by constraining searches and appropriately ordering search results. This section does not repeat that detailed discussion of different search techniques. Instead, it focuses on technical and practical requirements for delivering and accessing services that meet those requirements.

Several services described earlier in the guide provide technical ways to constrain searches for concepts to meet particular requirements.

- 4.2 Get a Concept, Description or Relationship introduced services that find concepts by identifier[1].
- 4.5 Get and Test Concept Subtypes and Supertypes includes consideration of services that find concepts based on subtype relationships to or from other concepts.
- 4.6 Get and Test Reference Set Membership includes services that find concepts that are members of a particular reference set.
- 4.7 Validate and Apply Expression Constraints describes the way that the above approaches to finding concepts can be combined.

This section indicates ways in which term searches can and should be combined with those constraints to enable effective searches for concepts relevant to a particular context of use.

Requirements and Options

Term Searches

High-performance term search services are essential for most practical applications of SNOMED CT. Search services also need to employ effective strategies for rationalizing, sorting and presenting search results in ways that facilitate selection of concepts most frequently used in a given context of use. Term searches are typically language or dialect specific. Therefore requests for searches should include appropriate language codes or language reference set identifiers[2].

Search Techniques and Search Strings

Section 4. Optimizing Searches in the *Search and Data Entry Guide* identifies a range of different search techniques that may be valuable. If a terminology server supports more than on search technique the service request will need to indicate the technique to be used.

In the simplest case a term search service should be able to take a term or phrase typed by a user and return matching results. For this purpose, the recommended default search technique is to search for words or parts of words in any order (see Search and Data Entry Guide 4.1.5. Search for Words within in Any Order). Searches using this technique should be to return concepts associated with terms that include words that match or begin with all the words or part words in the search string. The order of words need not be the same in the search string and the matched term and the search string need not include all the words in the term.

Specific data storage and development environments also support specific solutions for searching text. These include boolean searches (which require some words to be present and other words to be absent, natural language searches which may include words with similar meanings. In addition to these there are more general pattern matching approaches including the use of wildcard character and searches using regular expressions. While these techniques can be useful, end users should be able to search SNOMED CT effectively without being required to understand a specific technical syntax.

Search Result Filtering and Ordering

Section 5. Optimize Display of Search Results in the *Search and Data Entry Guide* identifies a range of options for filtering and ordering the results of a search. Filtering options include avoiding displaying multiple matching terms that refer to the same concept as well as ensuring that by default descriptions associated with inactive concept are not displayed (though there should be an option to include inactive concepts as these may be relevant to historical data). Ordering options include displaying the closest matches, shortest matching terms or most commonly used concepts at the top of the list of results.

Constrained Term Searches

Many practical use cases are best addressed by applying term searches to a constrained set of concepts. Searches that are appropriately constrained result in shorter lists of matching results and thus make it easier for a user to find the required concept. Constrained searches also reduce the risk of inadvertently selecting a concept in a hierarchy that is not appropriate to the data entry context. Thus this technique simplifies data entry and leads to improvements in data quality.

Example

Searching the entire content of SNOMED CT when looking for a diagnosis or reason for admission is likely to include many concepts that are are not valid diagnoses or reasons for admission. This can make if difficult for the user to find the concept that they need and, in some case, may lead to recording of concepts that have terms that do not represent the intended meaning. Examples of errors arising from failing to constrain searches include recording the names of substances and morphological abnormality rather than the name of symptom or disorder. The results of such errors can result in errors in reporting, analysis and decision support even in cases where the selected term may appear to capture the required meaning.

The recommended approach to constrained searches is to use expression constraints to specify the constraints. Ideally, the service should implement all the features of Expression Constraint Language (ECL). However, even if the service does not support some of the more advanced features, it should enable use of ECL to represent and combine subtype tests and reference set membership tests.

Services that do not support ECL should as a minimum support constraints based on reference set membership and the position of the concept in the subtype hierarchy. The recommended approach to testing the location of a concept in the hierarchy is the use of subtype tests. However, if these tests are not supported, a more limited approach that filters search results by hierarchy tags may be used.

✓ All those involved in developing or procuring SNOMED CT terminology services are strongly advised to refer to the detailed guidance on this topic in the SNOMED CT Search and Data Entry Guide .

Table 4.8-1: Essential and Recommended Functionality for Finding Concepts

Service Name and Status	**Input**	Output
Find concepts by term search REQUIRED	• Edition and version • Language code(s)[2] • Search string[3] • Search technique option (if required [4]) • Search filtering or sorting options (if supported[5])	A collection of matching terms each of which is linked to the appropriate concept.
Find concepts by constrained term search REQUIRED	• Edition and version • Language code(s) [2] • Search string[3] • Search technique option (if required[4]) • Search filtering or sorting options (if supported[5]) • An expression constraint[6]	A collection of matching terms that are applicable to concepts that conform to the specified constraints. Each term must be linked to the appropriate concept.

Interdependencies

Required By

- Other Services
  - 4.14 Validate Concept Definitions and Expressions
- Use Cases
  - 3.1 Explore and Review SNOMED CT
  - 3.2.3 EHR Data Entry
  - 3.2.2 EHR Data Entry Design
  - 3.4 EHR Reporting and Analytics

Depends On

- 4.1 Select Edition and Version
- 4.2 Get a Concept, Description or Relationship
- 4.3 Get Terms for a Concept
- 4.4 Get Definition of a Concept
- 4.5 Get and Test Concept Subtypes and Supertypes
- 4.6 Get and Test Reference Set Membership
- 4.7 Validate and Apply Expression Constraints

Service Examples

✓ The Snowstorm and FHIR examples are presented in plain text and URL encoded versions. Always use the "Encoded URL" when testing the example service requests. The plain text version is included to aid readability but using this version in a service request may result in errors. These errors result from characters that have to be encoded as they are not permitted in a URL (see IETF RFC1738).

Table 4.8-2: Snowstorm API

Service Name	API Call [7]	Result

Find concepts by term search	`GET [snowstorm]/snomed-ct/[branchPath]/concepts?activeFilter=true&term=[search-string]`
	Example 1
	`GET [snowstorm]/MAIN/2020-01-31/concepts?activeFilter=true&term=knee`
	Encoded URL
	`GET [snowstorm]/MAIN%2F2020-01-31/concepts?activeFilter=true&term=knee`
	Example 2.
	`GET [snowstorm]/MAIN/2020-01-31/concepts?activeFilter=true&term=alcohol`
	Encoded URL
	`GET [snowstorm]/MAIN%2F2020-01-31/concepts?activeFilter=true&term=alcohol`
	Example 3.
	`GET [snowstorm]/MAIN/2020-01-31/concepts?activeFilter=true&term=ren ston`
	Encoded URL
	`GET [snowstorm]/MAIN%2F2020-01-31/concepts?activeFilter=true&term=ren+ston`
		Returns a JSON representation of data related to the concepts that have terms that match the search string.
		The data returned for each concept includes:
		• All concept release file data[4]
		• The preferred term and fully specified name.
		Also returns the **total** number of concepts that match the search string.
		As some searches are matched by large numbers of concepts, this service is paged. Requests parameters include:
		• **limit** to restrict the number of concepts returned (default 50).
		• **offset** to specify the start in the results (in multiples of the limit).
		The results of the searches shown when applied to the 2020-01-31 International Edition were as follows:
		• **Example 1** returns the first 50 concept that match the search term "knee". It also return a total value of 1387 indicating that there are 1337 more matches that could be shown. If the word "x-ray" is added to the search term, this reduces the number of matches to 8. This is an example of how an additional term can provide refinement for a particular context of use. However, this does not necessarily include all radiographic procedures on the knee - for example "Computed tomography of knee" will not match.
		• **Example 2** returns the first 50 of 509 matches for the search term "alcohol". Of the first 50 concepts shown most related relate to the substance alcohol (26) and only 11 relate to disorders and findings.
		• **Example 3** uses a search term with two partial words "ren ston". This finds only 7 matches but even in this case, the search results contain concepts from five different SNOMED CT hierarchies (substance, specimen, disorder, procedure and situation).
		In all the above examples adding another word or two greatly reduces the results returned and reduces the distribution of the results across multiple hierarchies. However, when entering data for a given purpose it makes sense to limit the search results to concepts that can sensibly be applied to those contexts.

Find concepts by constrained term search

```
GET [snowstorm]/snomed-ct/
[branchPath]/concepts?
activeFilter=true&term=[searc
h-
string]&ecl=[expressionConstr
aint]
```

Example 4.

```
GET [snowstorm]/MAIN/
2020-01-31/concepts?
activeFilter=true&ecl=<363680
008|Radiographic imaging
procedure|&term=knee
```

Encoded URL

```
GET [snowstorm]/
MAIN%2F2020-01-31/concepts?
activeFilter=true&ecl=%3C3636
80008%7CRadiographic+imaging+
procedure%7C&term=knee
```

Example 5.

```
GET [snowstorm]/MAIN/
2020-01-31/concepts?
activeFilter=true&term=alcoho
l&ecl=<64572001|Disease|
```

Encoded URL

```
GET [snowstorm]/
MAIN%2F2020-01-31/concepts?
activeFilter=true&term=alcoho
l&ecl=%3C64572001%7CDisease%7
C
```

Example 6.

```
GET [snowstorm]/MAIN/
2020-01-31/concepts?
activeFilter=true&term=alcoho
l&ecl=<64572001|Disease| and
^450970008| General
Practice / Family Practice
reference set|
```

Constrained searches return JSON data for each of the matches in the same way as described above for the unconstrained searches.

Example 4 provides an example of how a search for x-ray procedures can be constrained so it is only necessary to enter the site and other related details. In this case the search term "knee" returns 40 matches. These include concepts such as "computed tomography of knee" which do not match "knee x-ray". The same constraint would simplifies all searches for radiology procedures and the constraint could also relaxed to include all imaging procedures.

Examples 5 and 6 provide an examples of how a search for a reason for admission might be constrained. In example 5 the constraint to disease restricts searches for alcohol or other drugs to concepts in the disorder hierarchy. This reduces the count of matches to 148. In example 6 a further constraint is added requiring the concept to be in the "General Practice / Family Practice reference set". The result of this is that only 21 matching concepts are returned. It is important to note that this is only an example, for practical in different clinical environments more specific reference sets should be considered.

Example 7 restricts the "ren ston" search to subtypes of disease and result in only 2 matches, which is a worryingly low number because there are certainly more SNOMED CT concepts that represent different disorders involving renal stones. The problem here is that many of the concepts do not have terms that match the search string. Instead, most related concepts use synonymous terms like "renal calculus" or "kidney stone".

> ✓ A useful option to allow the user to deal with this situation is for the client application to provide users with the option to expand the results to include the subtype children or descendants of the matching concepts. In the case illustrated by example 7, this option would return 18 additional concepts (subtype descendants of 95570007|Kidney stone| that did not match the search term.

<table>
<tr><td></td><td><i>Encoded URL</i>
GET [snowstorm]/MAIN%2F2020-01-31/concepts?activeFilter=true&term=alcohol&ecl=%3C64572001%7CDisease%7C+and+%5E450970008%7C+General+Practice+%2F+Family+Practice+reference+set%7C%0A
Example 7.
GET [snowstorm]/MAIN/2020-01-31/concepts?activeFilter=true&term=renston&ecl=<64572001|Disease|
<i>Encoded URL</i>
GET [snowstorm]/MAIN%2F2020-01-31/concepts?activeFilter=true&term=ren+ston&ecl=%3C64572001%7CDisease%7C</td><td></td></tr>
</table>

Table 4.8-3: FHIR API

Service Name	API Call [8]	Result

Find concepts by term search	In FHIR, the ValueSet/$expand operation may be used to search within a specified collection of codes.
	
	A text filter may be applied to FHIR requests to restrict the codes that are returned. The interpretation of this is delegated to the server in order to allow to determine the most optimal search approach for the context.
	
	Typical usage of this parameter includes functionality like:
	• using left matching e.g. "acut ast"
	• allowing for wild cards such as %, &, ?
	• searching on definition as well as display(s)
	• allowing for search conditions (and / or / exclusions)
	For further details, see http://hl7.org/fhir/valueset-operation-expand.html
	
	`GET [fhir]/ValueSet/$expand?url=[versionURI]?fhir_vs&count=10&filter=[search-string]`
	
	Example 1
	
	`GET [fhir]/ValueSet/$expand?url=http://snomed.info/sct/900000000000207008/version/20200131?fhir_vs&filter=knee`
	
	Encoded URL
	
	`GET [fhir]/ValueSet/$expand?url=http%3A%2F%2Fsnomed.info%2Fsct%2F900000000000207008%2Fversion%2F20200131%3Ffhir_vs&filter=knee`
	
	Example 2
	
	`GET [fhir]/ValueSet/$expand?url=http://snomed.info/sct/900000000000207008/version/20200131?fhir_vs&filter=ren ston`
	
	Encoded URL
		Returns a JSON representation of data related to the concepts that have terms that match the search string.
		
		The data returned for each concept includes:
		• The codesystem to which the returned concept belongs
		• The identifier (code) of the returned concept
		• A display term associated with the concept, e.g. the preferred term
		• All concept release file data[4]
		• The preferred term and fully specified name.
		Also returns the **total** number of concepts that match the search string.
		
		As some searches are matched by large numbers of concepts, restrictions may be applied to limit the number of returned concepts. Parameters include:
		• **count** to restrict the number of concepts returned.
		• **offset** to specify the start in the results (in multiples of the limit).
		Example 1 returns the first 10 concepts that match the search term "knee". As indicated by the first part of the url, the version requested is the January 2020 version of the International Edition. The total number of concepts returned is 1387.
		
		Example 2 uses the a search term with two partial words "ren ston". This finds only 7 matches but even in this case, the search results contain concepts from five different SNOMED CT hierarchies (substance, specimen, disorder, procedure and situation). However, please note that when using this FHIR service, the semantic tag (or FSN) is not provided in the response.

	`GET [fhir]/ValueSet/$expand?url=http://snomed.info/sct/900000000000207008/version/20200131?fhir_vs&filter=renston`	

<table>
<tr>
<td><b>Find concepts by constrained term search</b></td>
<td><code>GET [fhir]/ValueSet/$expand?url=[version]?fhir_vs=ecl/[expressionConstraint]&filter=[search-string]</code>
Example 3
<code>GET [fhir]/ValueSet/$expand?url=http://snomed.info/sct/900000000000207008/version/20200131?fhir_vs=ecl/<363680008&count=10&filter=knee</code>
<i>Encoded URL</i>
<code>GET [fhir]/ValueSet/$expand?url=http%3A%2F%2Fsnomed.info%2Fsct%2F900000000000207008%2Fversion%2F20200131%3Ffhir_vs%3Decl%2F%3C363680008&count=10&filter=knee</code>
Example 4
<code>GET [fhir]/ValueSet/$expand?url=http://snomed.info/sct/900000000000207008/version/20200131?fhir_vs=ecl/<64572001|Disease|&filter=alcohol</code>
<i>Encoded URL</i>
<code>GET [fhir]/ValueSet/$expand?url=http%3A%2F%2Fsnomed.info%2Fsct%2F900000000000207008%2Fversion%2F20200131%3Ffhir_vs%3Decl%2F%3C64572001%7CDisease%7C&filter=alcohol</code>
Example 5</td>
<td>Returns a JSON representation of data related to the concepts that have terms that match the search string.
The data returned for each concept includes:
• The codesystem to which the returned concept belongs
• The identifier (code) of the returned concept
• A display term associated with the concept, e.g. the preferred term
• All concept release file data[4]
• The preferred term and fully specified name.
Also returns the <b>total</b> number of concepts that match the search string.
As some searches are matched by large numbers of concepts, restrictions may be applied to limit the number of returned concepts. Parameters include:
• <b>count</b> to restrict the number of concepts returned.
• <b>offset</b> to specify the start in the results (in multiples of the limit).
<b>Example 3</b> provides an example of how a search for x-ray procedures can be constrained so it is only necessary to enter the site and other related details. In this case the search term "knee" returns 40 matches. These include concepts such as "computed tomography of knee" which do not match "knee x-ray". The same constraint would simplifies all searches for radiology procedures and the constraint could also relaxed to include all imaging procedures.
<b>Examples 4 and 5</b> provide an examples of how a search for a reason for admission might be constrained. In example 4, the constraint to disease restricts searches for alcohol or other drugs to concepts in the disorder hierarchy. This reduces the count of matches to 148. In example 5 a further constraint is added requiring the concept to be in the "General Practice / Family Practice reference set". The result of this is that only 21 matching concepts are returned. It is important to note that this is only an example, for practical in different clinical environments more specific reference sets should be considered.</td>
</tr>
</table>

	GET [fhir]/ValueSet/$expand? url=http://snomed.info/sct/ 900000000000207008/version/ 20200131?fhir_vs=ecl/ <64572001\|Disease\|and ^450970008\| General Practice / Family Practice reference set\|&filter=alcohol *Encoded URL* GET [fhir]/ValueSet/$expand? url=http%3A%2F%2Fsnomed.info% 2Fsct%2F900000000000207008%2F version%2F20200131%3Ffhir_vs% 3Decl%2F%3C64572001%7CDisease %7Cand+ %5E450970008%7C+General+Pract ice+ %2F+Family+Practice+reference +set%7C&filter=alcohol	

Table 4.8-4: MySQL Example Database

Service Name	SQL Query [9]	Result

Find concepts by term search	SELECT s.conceptId 'id',s.term 'term', f.term 'FSN'
	    FROM snap_term_search_active s
	    JOIN snap_fsn f on s.conceptId=f.conceptId
	    WHERE MATCH (s.term) AGAINST ('[search-string]' IN BOOLEAN MODE)
	    ORDER BY LENGTH(s.term)
	for example
	SELECT s.conceptId 'id',s.term 'term', f.term 'FSN'
	    FROM snap_term_search_active s
	    JOIN snap_fsn f on s.conceptId=f.conceptId
	    WHERE MATCH (s.term) AGAINST ('knee' IN BOOLEAN MODE)
	    ORDER BY LENGTH(s.term)
	SELECT s.conceptId 'id',s.term 'term', f.term 'FSN'
	    FROM snap_term_search_active s
	    JOIN snap_fsn f on s.conceptId=f.conceptId
		Returns the conceptId, term found and fully specified name. Ordering the results so that the closest matches (shortest matching terms) appear first in the list.

	`WHERE MATCH (s.term) AGAINST ('+renal +stone' IN BOOLEAN MODE)`
	`ORDER BY LENGTH(s.term)`	

Find concepts by constrained term search	`CALL snap_SearchPlus('[search-string]','[simple-constraint]');`
	for example
	`CALL snap_SearchPlus('knee','<363680008');`
	`CALL snap_SearchPlus('+alcohol intoxication','<64572001');`
		The searchPlus procedure allows a search string to be combined with a simple subtype constraint. This returns the conceptId and term for each match found.
		This current searchPlus procedure in the example MySQL database does not support more complex constraint expressions.

Footnotes

[1] It should never be necessary for an end-user to enter a concept identifier. However, there are practical situations in which concept identifiers may be available to be used to find a specific concept or set of concepts. These situations include coded data in existing records or messages; templates, pick lists and other user interface controls that are bound to specific concept identifiers. If concept identifiers are available they provide a direct way to find data about specific concepts.

[2] One or more languages and/or dialects should be specified for searches. If more than one language or dialect is specified, the search should return matching terms that are acceptable in any specified language or dialect.

[3] The nature of the search string may vary according the search techniques supported and any options selected (see Search Techniques and Search Strings).

[4] The search technique needs to be specified if the service supports different techniques (see Search Techniques and Search Strings).

[5] If the service supports alternative ways of filtering or ordering results, these options chosen need to be specified (see Search Result Filtering and Ordering).

[6] As noted in Constrained Term Searches, the use of expression constraints is recommended. However, other representations of constraints may be used if the service does not support expression constraint language.

[7]

In the Snowstorm service requests [snowstorm] should be replaced by the URL to the Snowstorm server endpoint.

[8]
In the FHIR service requests [fhir] should be replaced by the URL to the FHIR terminology server endpoint. FHIR® is a registered trademarks of HL7 (www.hl7.org).

[9]
The SNOMED CT MySQL example database is **not** designed as a terminology server and is **not** intended for use in a live system . It is referenced in this guide as an illustration that some readers may find helpful. For more information about the SNOMED CT example database see the SNOMED CT - SQL Practical Guide. For instructions on how to build the example database refer to Appendix A: Building the SNOMED CT Example Database.

4.9 Identify Changes to the Terminology

Overview

SNOMED CT release files include data that allows tracking of all changes between different versions of any edition of the terminology. In addition to a unique id, each row in every release file includes the following columns effectiveTime, active and moduleId. The *effectiveTime* column shows when the row became the effective version of that component or reference set member, either as a new component or as revision of the previous version of that component. The *active* column indicates whether the component is still intended for active use. The *moduleId* identifies the module that the component belongs to. A change to the *moduleId* indicates that a component has been moved to a different module. The new module may be another module maintained and distributed by the same organization. However, in some cases, a component may be moved to a module that is maintained and distributed by another organization.

Before an organization updates to use a newer SNOMED CT version, it is important to identify significant changes that have occurred between the two versions. As a minimum, it is essential to identify the inactivation of any concepts and descriptions that are used in data entry templates, picklists, reporting and analytics queries. Additionally, where reference sets are used to assist or constrain data entry, changes to the membership of those reference sets should be reviewed. A more extensive review covering changes to other reference set may also be necessary where these are used as part of data entry, reporting or analysis.

This section is concerned with services that identify and categorize these changes. A later section of the guide (4.11 Get History Data) addresses additional services that provide more information about some of these changes.

Requirements and Options

To identify changes, terminology services must be able to provide the following information about data in the full release of the new version:

1. Identify components and reference set members[1] with an effectiveTime greater than the date of the previously installed version[2].
2. For each component identified by step 1:
    a. Check the active status in the updated release (updated=**active** or updated=**inactive**)
    b. Check if the component was present in the previous release and if so whether it was active or inactive. (previous=**active**, previous=**inactive** or previous=**none**)
3. Based on the results of step 2, use Table 4.9-1 Terminology Component Update Types to determine:
    a. The type of update that has occurred;
    b. The actions that need to be considered.

The required services are listed in Table 4.9-2.

Table 4.9-1: Terminology Component Update Types

Previous	Updated	Update Type	Comment	Actions to Consider
none	**active**	**Addition**	This component has been added in since the previous release.	**Concept or description**: Consider including in data entry templates, picklists, etc. **Reference set member**: May need to consider impact of addition if the reference set is used in constraints for data entry or queries used for reporting.
active	**active**	**Change**	The component has been changed but remains active. Changes have been made to one or more of the other mutable properties of the component. In some cases, the only change made may have been to the moduleId indicating a reorganization of components within the release package.	Check the specific nature of change and consider if this has any impact.
active	**inactive**	**Inactivation**	The component has been inactivated since the previous release. • For inactivated concepts and descriptions, information about the reason for inactivation should be reviewed. • For inactivated concepts, historical associations linking the inactive concept to relevant active concepts should also be reviewed.	**Concept**: Review and replace any direct use in data entry templates, picklists, constraints and queries. Use history data (see 4.11 Get History Data) to determine reason for inactivation and to find associations with potential active replacement concepts. Review impact on reporting or existing records using the concept. **Description**: Review and replace any direct use in data entry templates, picklists. Use history data (see 4.11 Get History Data) to determine reason for inactivation **Reference set member**: Review the impact of the change on data entry templates, picklists, constraints, queries, mapping, etc.
			The following situations are unusual but may sometimes occur	
inactive	active	**Reactivation**	The component has been reactivated. This may represent a correction of a previous inactivation.	If the component is a concept or description that caused an issue when inactivated, consider reversing changes made when it was inactivated.
inactive	inactive	**Remains inactive**	This component remains inactive but has changed in some way. This is an unusual situation, it may indicate a change in some other data since inactivation.	No action required.
none	inactive	**Inactivated addition**	If this occurs, the likely cause is an addition in an intervening release, which has subsequently inactivated. From a practical perspective this is equivalent to no change. However, the concept may occur in a communication from another organization that used a version in which the component way active.	**Concept or description**: No action required locally, but be aware that communicated data might include this now inactive concept or description. This is possible in communicated data from a system that used an intermediate release in which the component was active.

Table 4.9-2: Services Required

Service Name and Status	Input	Output
Get components or reference set members that have changed since a specified previous release **REQUIRED**	• Edition • Version release date/time (snapshot of components and refset members at this date/time represents the updated state) • Previous version release date/time (snapshot of components and refset members at this date/time represents the previous state) • Component type or reference set type • Optional: refsetId • Optional: Language/dialect[3]	• Identifiers of components or reference set members with the following characteristics: ▪ specified component or reference set type ▪ effectiveTime greater than previous release date/time (note that if only date is specified then component with any effectiveTime on that date must be excluded) • Indication of the update type (see Table 4.9-1). The update type indication should be applied either to: a. the set of identifiers to which it applies; or b. individually to each identifier. Optionally other data associated with each changed component or reference set member could also be returned. However, requirements for additional data vary depending on the update type and the type of component or reference set. A service that returns sets of identifiers grouped by update type, and component or reference set type offers a more flexible solution. It allows additional data to be retrieved selectively by other services to support appropriate review of the changes.

Interdependencies

Required By

- Other Services
  - 4.11 Get History Data
- Use Cases
  - 3.7.3 Manage Impact of Changes on EHR Applications
  - 3.7.4 Manage Impact of Changes on Extensions

Depends On

- 4.1 Select Edition and Version (including support for selecting two versions for comparison)
- 4.2 Get a Concept, Description or Relationship
- 4.3 Get Terms for a Concept
- 4.4 Get Definition of a Concept
- 4.5 Get and Test Concept Subtypes and Supertypes
- 4.6 Get and Test Reference Set Membership

Service Examples

✓ The Snowstorm and FHIR examples are presented in plain text and URL encoded versions. Always use the "Encoded URL" when testing the example service requests. The plain text version is included to aid readability but using this version in a service request may result in errors. These errors result from characters that have to be encoded as they are not permitted in a URL (see IETF RFC1738).

Table 4.9-3: Snowstorm API

Service Name	API Call[4]	Result
Get components or reference set members that have changed since a specified previous release	N/A	This service is not currently supported. ⚠ When Snowstorm is running with write access enabled it is possible to export SNOMED CT content selected by a range between two effective times. This process could be adapted to allow the relevant information to be processed in an application to identify the components and reference set members that have been updated. However, this would require the client application to process the selected data after it has been exported in the release file format.

Table 4.9-4: FHIR API

Service Name	API Call[5]	Result
Get components or reference set members that have changed since a specified previous release	N/A	This service is not supported.

Table 4.9-5: MySQL Example Database

Service Name	SQL Query [6]	Result

Get components or reference set members that have changed since a specified previous release	`SELECT *`
	`    FROM [snapshotTableName]`
	`        WHERE effectiveTime >`
	`[previousVersionDate];`
	Example 1
	`SELECT *`
	`    FROM `snap_concept``
	`        where `effectiveTime``
	`> '2019-07-31';`
	Example 2
	`SELECT *`
	`    FROM `snap_description``
	`        where `effectiveTime``
	`> '2017-01-31';`
	Example 3
	`CALL setSnapshotTime(1,`
	`'2019-07-31);`
	`SELECT *`
	`    FROM snap1_refset_simple`
	`        WHERE effectiveTime >`
	`'2018-07-31';`
		Returns all rows in the specified snapshot view of the named release file that have an effectiveTime is later than the specified previous version date[7]. The snapshot time for the snapshot view determines the date of the current version.
		Example 1 returns all rows in the current snapshot of the concept release file that have an effectiveTime after 2019-07-31.
		Example 2 returns all rows in the current snapshot of the description release file that have an effectiveTime after 2017-01-31.
		Example 3 sets a specific snapshot time 1 as (2019-07-31). It then returns rows from the simple reference set table in that specific snapshot view that have an effectiveTime after 2018-07-31.

Footnotes

[1] The significance of changes to membership of reference sets depends on the extent to which they are used by a particular application. For example, a change to language reference set membership may mean that a description is no longer acceptable in the language or dialect used in by an application. Similarly, a change to a the membership of a simple reference set used to constrain a searches, reports or analytics may alter the results of those actions.

[2] A delta release file may be available to provide information about changes since the previous release. However, in cases where an organization does not update the version they are using regularly, there may not be a single set of *delta release* files covering all the changes made between the version in use and the current version.

[3] Language and/or dialect should be specified if the service returns terms associated with referenced concepts.

[4] In the Snowstorm service requests [snowstorm] should be replaced by the URL to the Snowstorm server endpoint.

[5] In the FHIR service requests [fhir] should be replaced by the URL to the FHIR terminology server endpoint. FHIR® is a registered trademarks of HL7 (www.hl7.org).

[6] The SNOMED CT MySQL example database is not designed as a terminology server and is not intended for use in a live system . It is referenced in this guide as an illustration that some readers may find helpful. For more information

about the SNOMED CT example database see the SNOMED CT - SQL Practical Guide. For instructions on how to build the example database refer to Appendix A: Building the SNOMED CT Example Database.

7
Note that to avoid issues if hours, minutes or seconds are included in previousVersionDate, the value should be set as 23:59:59 on the required date.

4.10 Get Data from a Reference Set

Overview

Reference set files provide a standard format for maintaining and distributing a set of references to SNOMED CT components.

The services required to identify the members of a reference set are described in Section 4.6 Get and Test Reference Set Membership. This section describes general requirements for services that are able to access additional data stored in reference sets.

The following services, described in earlier sections, require access to a single data value from each reference set row:

- 4.3 Get Terms for a Concept requires access to the acceptabilityId in a language reference set to determine the preferred and acceptable terms for a concept in a specified language or dialect.
- 4.4 Get Definition of a Concept requires access to the owlExpression in the OWL axiom reference set to get the stated view of a concept definition.

The following services described in the next few sections, require access to one or more data values from each reference set row:

- 4.11 Get History Data uses historical reference set data to determine the reason for inactivation and to identify related active concepts.
- 4.12 Get Mapping Data uses mapping reference set data to access maps to or from SNOMED CT concepts or expressions.
- 4.13 Get Concept Model Rules uses MRCM reference set data to access concept model domain, attribute and range related rules.

Other reference set types that are not specifically documented in this section of the guide also contain additional data to which a client application may require access. Furthermore when new reference sets are introduced, there will be requirements to access additional data items in those reference sets. The service described in this section provides a general solution by providing access to all additional data associated with a reference set member in any type of reference set.

Requirements and Options

The required services are listed in Table 4.10-1.

Table 4.10-1: Services Required

Service Name and Status	Input	Output
Get data from a reference set for a specific referenced component **REQUIRED**	• Edition and version • refsetId for the reference set from which data is to be extracted. ▪ Optional alternatives: a set of refsetIds or an expression constraint for a group of related reference sets. • Optional: referencedComponentId • Optional parameters to select items within a particular type of reference set for example: ▪ mapTarget (in a mapping reference set) ▪ targetComponentId (in an association or ordered association reference set)	Data from specified rows in the identified reference set(s). • Note that the available data depends on the reference set type

Interdependencies

Required By

- Other Services
  - 4.3 Get Terms for a Concept
  - 4.4 Get Definition of a Concept
  - 4.11 Get History Data
  - 4.12 Get Mapping Data
  - 4.13 Get Concept Model Rules

Depends On

- 4.1 Select Edition and Version
- 4.6 Get and Test Reference Set Membership

Service Examples

✓ The Snowstorm and FHIR examples are presented in plain text and URL encoded versions. Always use the "Encoded URL" when testing the example service requests. The plain text version is included to aid readability but using this version in a service request may result in errors. These errors result from characters that have to be encoded as they are not permitted in a URL (see IETF RFC1738).

Table 4.10-2: Snowstorm API

Service Name	API Call[2]	Result

Get data from a reference set for a specific referenced component	`GET [snowstorm]/[branchPath]/members?active=true&referenceSet=[refsetId-or-ecl][&refsetComponentId=[componentId][&other-refsetType-specific-parameters]`
	Example 1
	`GET [snowstorm]/MAIN/2020-01-31/members?active=true&referenceSet=900000000000523009|POSSIBLY EQUIVALENT TO association reference set|&referencedComponentId=203004`
	Encoded URL
	`GET [snowstorm]/MAIN%2F2020-01-31/members?active=true&referenceSet=900000000000523009%7CPOSSIBLY+EQUIVALENT+TO+association+reference+set%7C&referencedComponentId=203004`
	Example 2
	`GET [snowstorm]/MAIN/2020-01-31/members?active=true&referenceSet=<900000000000522004|Historical association reference set|`
	Encoded URL
	`GET [snowstorm]/MAIN%2F2020-01-31/members?active=true&referenceSet=%3C900000000000522004%7CHistorical+association+reference+set%7C`
	Example 3
		Returns a JSON representation of data from members of the specified reference set that match the specified criteria.
		The data returned for each matching reference set row includes:
		• Data from the common columns present in all reference sets.
		• The referencedComponentId
		  ▪ If the referencedComponentId refers to a concept, details of the concept including its fully specified name and preferred term are also included.
		• An additionalFields object containing specific data fields associated with the particular reference set type.
		Also returns the **total** number of reference set row that match the constraint.
		As some requests for reference set members return a large numbers of row, this service is paged. Requests parameters include:
		• **limit** to restrict the number of concepts returned (default 50).
		• **offset** to specify the start in the results (in multiples of the limit).
		Example 1 shows a search for data about a specific inactive component in a specific association reference set. This returns two rows each of which includes a targetComponentId referring to an active concept that represents a possible meaning of the inactive concept.
		Example 2 provides an example in which rows from all the historical association reference sets are returned.
		Example 3 demonstrates that it is also possible to specify a targetComponentId value when searching for members of reference sets that contains this data item. Similar options are also supported for searches using some other data items specific to certain types of reference set (e.g. mapTarget, owlExpression.conceptId).

<table>
<tr><td></td><td>GET [snowstorm]/MAIN/2020-01-31/members?active=true&referenceSet=<900000000000522004|Historical association reference set|&targetComponent=140004
<em>Encoded URL</em>
GET [snowstorm]/MAIN%2F2020-01-31/members?active=true&referenceSet=%3C900000000000522004%7CHistorical+association+reference+set%7C&targetComponent=140004</td><td></td></tr>
</table>

Table 4.10-3: FHIR API

Service Name	API Call[3]	Result

Get data from a reference set for a specific referenced component	The FHIR TS API supports the retrieval of targets for specific SNOMED CT reference sets. Please refer to this document for detailed guidance: https://www.hl7.org/fhir/snomedct.html. Thus, the ConceptMap/$translate operation enables the retrieval of targets for a specific referenced component.
	`GET [fhir]/ConceptMap/`
	`$translate?code=[componentId]`
	`&system=http://snomed.info/`
	`sct &source=http://`
	`snomed.info/sct?fhir_vs`
	`&target=http://snomed.info/`
	`sct?fhir_vs &url=[version]?`
	`fhir_cm=[refesetId]`
	for example
	`GET [fhir]/ConceptMap/`
	`$translate?code=203004`
	`&system=http://snomed.info/`
	`sct &source=http://`
	`snomed.info/sct?fhir_vs`
	`&target=http://snomed.info/`
	`sct?fhir_vs &url=http://`
	`snomed.info/sct/`
	`900000000000207008/version/`
	`20200131?`
	`fhir_cm=900000000000523009`
		Returns a JSON representation of data about each of the target components.
		The data returned for each concept includes:
		• **boolean:** True if the concept could be translated successfully. The value can only be true if at least one returned match
		• **match:** Each match represents data for the map or associated target. Note that there may be multiple matches, where each element represents a mapTarget. For each mapTarget, following data is provided
		• **system**: the codesystem of the mapTarget
		• **code:** The identifier of the mapTarget
		Example 1 shows a search for data about a specific inactive component in a specific association reference set. This returns two rows each of which includes a targetComponentId referring to an active concept that represents a possible meaning of the inactive concept.

Table 4.10-4: MySQL Example Database

Service Name	SQL Query [4]	Result

Get data from a reference set for a specific referenced component	SELECT * FROM
	 snap_refset_[refsetType]
	WHERE refsetId=[refsetId]
	    [AND other-refset-type-
	specific-criteria];
	Example 1
	SELECT * FROM
	 snap_refset_association
	    WHERE
	 refsetId=900000000000523009
	    AND active=1 AND
	 referencedComponentId=203004;
	Example 2
	SELECT * FROM
	 snap_refset_association
	    WHERE refsetId IN
	 (900000000000523009)
	    AND active=1 AND
	 referencedComponentId=203004
	UNION
	SELECT * FROM
	 snap_refset_attributevalue
	    WHERE refsetId IN
	 (900000000000489007)
	    AND active=1 AND
	 referencedComponentId=203004;
	Example 3
	SELECT * FROM
	 snap_refset_association
	WHERE refsetId IN
	(900000000000523009,900000000000489007)
	    AND active=1 AND
	 targetComponentId=140004;
		Returns a rows of data from the relevant reference set determined by the specified criteria.
		The data returned for each matching reference set row includes:
		- Data in the common columns present in all reference sets.
		- Additional data columns associated with the particular reference set type.
		Also returns the **total** number of reference set row that match the constraint.
		Example 1 shows a search for data about a specific inactive component in a specific association reference set. This returns two rows each of which includes a targetComponentId referring to an active concept that represents a possible meaning of the inactive concept.
		Example 2 provides an example in which rows from two an historical association reference sets and the concept inactivation attribute value reference set are returned for a specified referencedComponentId [5]
		Example 3 demonstrates that it is also possible to specify a targetComponentId value when searching for members of reference sets that contains this data item. Similar options are also supported for searches using some other data items specific to certain types of reference set (e.g. mapTarget).

```
SELECT * FROM
 snap_refset_association WHERE
 refsetId IN
 (900000000000523009,
900000000000524003,
900000000000525002,
900000000000526001,
900000000000527005,
900000000000528000,
900000000000529008,
900000000000530003,
900000000000531004)
    AND active=1 AND
 targetComponentId=140004;
```

Footnotes

[1] Language and/or dialect should be specified if the service returns terms associated with referenced concepts.

[2] In the Snowstorm service requests [snowstorm] should be replaced by the URL to the Snowstorm server endpoint.

[3] In the FHIR service requests [fhir] should be replaced by the URL to the FHIR terminology server endpoint. FHIR® is a registered trademarks of HL7 (www.hl7.org).

[4] The SNOMED CT MySQL example database is not designed as a terminology server and is not intended for use in a live system . It is referenced in this guide as an illustration that some readers may find helpful. For more information about the SNOMED CT example database see the SNOMED CT - SQL Practical Guide. For instructions on how to build the example database refer to Appendix A: Building the SNOMED CT Example Database.

[5] To keep the union query in SQL Example 2, as simple as possible, the column names have not been specified for the attribute value part of the query. As a result the valueId from that row is in the column labelled targetComponentId.

4.11 Get History Data

Overview

With each new release version of SNOMED CT there are requirements to manage changes in the terminology. One key element of this was covered by the requirements in section 4.9 Identify Changes to the Terminology. However, in addition to this there is a requirement access data that indicates the reasons for each of those changes and data that, where possible, provides links to active content that replaces components that have been made inactive. This data is distributed in historical reference sets.

An historical reference set is a reference set that provides information about concepts or descriptions that have been inactivated.

- There are two types of *historical reference sets*:
    - **Error rendering macro 'sp-plaintextbody-link'**

Conversion context did not contain original content entity.

which indicate the reason for inactivation of a particular component.

- **Error rendering macro 'sp-plaintextbody-link'**

  Conversion context did not contain original content entity.

which associate inactive concepts with one or more active concepts that represent the same, similar or a possible meaning of an inactive concept.

Requirements and Options

As noted in section 4.9 Identify Changes to the Terminology it is important to identify concepts and descriptions that have been inactivated since an earlier version. Once this has been done:

- Information about the reasons for inactivation of each description should be accessed from the 900000000000490003 |Description inactivation indicator attribute value reference set|
- Information about the reasons for inactivation of each concept should be accessed from the 900000000000489007 |Concept inactivation indicator attribute value reference set|
- Associations between each inactivated concepts and active concepts which represent similar meanings must be accessed from one of the < 900000000000522004 |Historical association reference set| subtypes.

The required services are listed in Table 4.11-1.

Table 4.11-1: Services Required

Service Name and Status	Input	Output
Get reason for description inactivation **REQUIRED**	• Edition and version • Identifier of an inactive description • Optional: Language/dialect[1]	Identifier and term representing the reason for inactivation of the description. For interpretation of the reasons for inactivation see 5.2.3.1 Component Inactivation Reference Sets.
Get reason for concept inactivation **REQUIRED**	• Edition and version • Identifier of an inactive concept • Optional: Language/dialect[1]	Identifier and term representing the reason for inactivation of the concept. For interpretation of the reasons for inactivation see 5.2.3.1 Component Inactivation Reference Sets.
Get historical associations between an inactive concept and one or more active concepts **REQUIRED**	• Edition and version • Identifier of an inactive concept • Optional: Language/dialect[1]	Identifiers and term(s) of related active concept(s) and the nature of the historical association between the inactive and active concept. For a description of each of the types of association see 5.2.5.1 Historical Association Reference Sets.

Interdependencies

Required By

- **Use Cases**
  - 3.7.3 Manage Impact of Changes on EHR Applications
  - 3.7.4 Manage Impact of Changes on Extensions

Depends On

- 4.1 Select Edition and Version
- 4.9 Identify Changes to the Terminology
- 4.10 Get Data from a Reference Set

Service Examples

The Snowstorm and FHIR examples are presented in plain text and URL encoded versions. Always use the "Encoded URL" when testing the example service requests. The plain text version is included to aid readability but using this version in a service request may result in errors. These errors result from characters that have to be encoded as they are not permitted in a URL (see IETF RFC1738).

Table 4.11-2: Snowstorm API

Service Name	**API Call**[2]	**Result**

<table>
<tr>
<td><b>Get reason for description inactivation</b></td>
<td>GET [snowstorm]/[branchPath]/members?active=true&referenceSet=900000000000490003|Description inactivation indicator attribute value reference set |&referencedComponentId=[inactiveDescriptionId]
for example
GET [snowstorm]/MAIN%2F2020-01-31/members?active=true&referenceSet=900000000000490003|Description inactivation indicator attribute value reference set |&referencedComponentId=78334016
<i>Encoded URL</i>
GET [snowstorm]/MAIN%2F2020-01-31/members?active=true&referenceSet=900000000000490003%7CDescription+inactivation+indicator+attribute+value+reference+set+%7C&referencedComponentId=78334016</td>
<td>Returns a JSON representation of data related to the specified description.
A single reference set member is returned and the targetComponentId refers to a concept that indicates the reason for inactivation of the concept:
- The referencedComponentId returned is the inactive concept to which the inactivation reason applies.
- The additionalFields / valueId refers to the concept that indicates the reason for inactivation.
    - The preferred term and fully specified name associated with the valueId describe the reason for inactivation. These terms can be looked up using Snowstorm services listed in 4.3 Get Terms for a Concept: Table 2 Snowstorm API.</td>
</tr>
</table>

<table>
<tr><td></td><td></td><td></td></tr>
<tr>
<td><b>Get reason for concept inactivation</b></td>
<td>GET [snowstorm]/[branchPath]/members?active=true&referenceSet=900000000000489007|Concept inactivation indicator attribute value reference set|&referencedComponentId=[inactiveConceptId]
for example
GET [snowstorm]/MAIN/2020-01-31/members?active=true&referenceSet=900000000000489007|Concept inactivation indicator attribute value reference set|&referencedComponentId=20559007
<i>Encoded URL</i>
GET [snowstorm]/MAIN%2F2020-01-31/members?active=true&referenceSet=900000000000489007%7CConcept+inactivation+indicator+attribute+value+reference+set%7C&referencedComponentId=20559007</td>
<td>Returns a JSON representation of data related to the specified concept.
A single reference set member is returned and the targetComponentId refers to a concept that indicates the reason for inactivation of the concept:
• The referencedComponentId returned is the inactive concept to which the inactivation reason applies.
• The additionalFields / valueId refers to the concept that indicates the reason for inactivation.
▪ The preferred term and fully specified name associated with the valueId describe the reason for inactivation. These terms can be looked up using Snowstorm services listed in 4.3 Get Terms for a Concept: Table 2 Snowstorm API.</td>
</tr>
</table>

<table>
<tr><th></th><th></th><th></th></tr>
<tr>
<td><b>Get historical associations between an inactive concept and one or more active concepts</b></td>
<td><code>GET [snowstorm]/[branchPath]/members?active=true&referenceSet=<900000000000522004|Historical association reference set|&referencedComponentId=[inactiveConceptId]</code>
for example
<code>GET [snowstorm]/MAIN/2020-01-31/members?active=true&referenceSet=<900000000000522004|Historical association reference set|&referencedComponentId=20559007</code>
<i>Encoded URL</i>
<code>GET [snowstorm]/MAIN%2F2020-01-31/members?active=true&referenceSet=%3C900000000000522004%7CHistorical+association+reference+set%7C&referencedComponentId=20559007</code></td>
<td>Returns a JSON representation of the historical association(s) of the specified inactive concept.
Each reference set member returned represents a historical association of the inactive concept with an active concept:
• The refsetId indicates the specific historical reference set(s) in which associations are found. This represents the nature of the association.
▪ The preferred term and fully specified name associated with the refsetId describe the association. These terms can be looked up using Snowstorm services listed in 4.3 Get Terms for a Concept: Table 2 Snowstorm API.
▪ If required, the refsetId returned can looked up using the get concept service to find the preferred term or fully specified name of the association reference set.
• The referencedComponentId returned is the inactive concept to which the association applies.
• The additionalFields / targetComponentId represents associated active concept.
• In the case of an ambiguous concept, each reference set member represents one possible meaning of the inactive concept.</td>
</tr>
</table>

Table 4.11-3: FHIR API

Service Name	API Call[3]	Result
Get reason for description inactivation	N/A	The FHIR TS API does not provide a service for this purpose
Get reason for concept inactivation	N/A	The FHIR TS API does not provide a service for this purpose

Get historical associations between an inactive concept and one or more active concepts	The FHIR TS API supports retrieval of targets for specific SNOMED CT reference sets. Please refer to this document for detailed guidance: https://www.hl7.org/fhir/snomedct.html. Thus, the ConceptMap/$translate operation enables the retrieval of targets for a specific referenced component.
	`GET [fhir]/ConceptMap/$translate?code=[componentId] &system=http://snomed.info/sct &source=http://snomed.info/sct?fhir_vs &target=http://snomed.info/sct?fhir_vs &url=[version]?fhir_cm=[refesetId]`
	for example,
	`GET [snowstorm]/ConceptMap/$translate?code=134811001&system=http://snomed.info/sct&source=http://snomed.info/sct?fhir_vs&target=http://snomed.info/sct?fhir_vs&url=http://snomed.info/sct?fhir_cm=900000000000527005`
	Encoded URL
	`GET [snowstorm]/ConceptMap/$translate?code=134811001&system=http%3A%2F%2Fsnomed.info%2Fsct&source=http%3A%2F%2Fsnomed.info%2Fsct%3Ffhir_vs&target=http%3A%2F%2Fsnomed.info%2Fsct%3Ffhir_vs&url=http%3A%2F%2Fsnomed.info%2Fsct%3Ffhir_cm%3D900000000000527005`
		Returns a JSON representation of data about each of the target components.
		The data returned for each concept includes:
		• **boolean:** True if the concept could be translated successfully. The value can only be true if at least one returned match
		• **match:** Each match represents data for the map or associated target. Note that there may be multiple matches, where each element represents a mapTarget. For each mapTarget, following data is provided
		• **system**: the codesystem of the mapTarget
		• **code:** The identifier of the mapTarget
		The example shows the retrieval of the active replacement for the inactive concept 134811001 |Anaesthetist (occupation)|. The historical association reference set is the 900000000000527005 |SAME AS association reference set (foundation metadata concept)|, and the request result shows the concept Id of the active concept 88189002 |Anesthesiologist (occupation)|.

Table 4.11-4: MySQL Example Database

Service Name	SQL Query[4]	Result

<table>
<tr>
<td><b>Get reason for description inactivation</b></td>
<td>
<pre>
CALL setDeltaRange(1,
[deltaStartTime],
[deltaEndTime]);
SELECT * FROM
 delta_inactive_descriptions;
</pre>
for example
<pre>
CALL setDeltaRange(1,'2019-07-
31', '2020-01-31');
SELECT * FROM
 delta_inactive_descriptions;
</pre>
</td>
<td>
Returns a row of data for each description inactivated after the deltaStartTime up to and including the deltaEndTime. Each row contains the following columns:
<table>
<tr><th>Columns</th><th>Description</th></tr>
<tr><td><b>id</b></td><td>Identifier of the inactivated description.</td></tr>
<tr><td><b>effectiveTime</b></td><td>Effective time of the row that inactivated the description</td></tr>
<tr><td><b>active</b></td><td>Active state of description. Should always be zero (0)</td></tr>
<tr><td><b>conceptid</b></td><td>Identifier of the concept to which the description applies</td></tr>
<tr><td><b>term</b></td><td>Term of the inactivated description</td></tr>
<tr><td><b>concept_fsn</b></td><td>The fully specified name of the concept.</td></tr>
<tr><td><b>concept_active</b></td><td>Active state of the concept.</td></tr>
<tr><td><b>reason</b></td><td>The reason for inactivation represented by the preferred term associated with the valueId concept in the relevant row of the description inactivation reference set.</td></tr>
</table>
</td>
</tr>
</table>

<table>
<tr>
<td><b>Get reason for concept inactivation and historical associations between each inactive concept and one or more active concepts</b></td>
<td>
CALL setDeltaRange(1,
[deltaStartTime],
[deltaEndTime]);
SELECT * FROM
 delta1_inactive_concepts;

for example

CALL setDeltaRange(1,'2019-07-
31', '2020-01-31');
SELECT * FROM
 delta1_inactive_concepts;
</td>
<td>
Returns one or more rows of data for each concept inactivated after the deltaStartTime up to and including the deltaEndTime. Each row contains the following columns:

id, effectiveTime, active, definitionStatusId, FSN, reason, assoc_type, ref_conceptId, ref_concept_FSN

<table>
<tr><th>Columns</th><th>Description</th></tr>
<tr><td><b>id</b></td><td>Identifier of the inactivated concept</td></tr>
<tr><td><b>effectiveTime</b></td><td>Effective time of the row that inactivated the concept</td></tr>
<tr><td><b>active</b></td><td>Active state of concept. Should always be zero (0)</td></tr>
<tr><td><b>definitionStatusId</b></td><td>Definition status of the inactivated concept</td></tr>
<tr><td><b>FSN</b></td><td>Fully specified name of the inactivated concept</td></tr>
<tr><td><b>reason</b></td><td>The reason for inactivation represented by the preferred term associated with the valueId concept in the relevant row of the concept inactivation reference set.</td></tr>
<tr><td><b>assoc_type</b></td><td>The preferred term for the name of the association reference set containing the association between the inactive and active concept.</td></tr>
<tr><td><b>ref_conceptId</b></td><td>The identifier of the active concept (or one of the active concepts) with which the inactive concept is associated.</td></tr>
<tr><td><b>ref_concept_FSN</b></td><td>The fully specified name of the active concept with which the inactive concept is associated.</td></tr>
</table>
Note that in the case of a concept that has been inactivated due to ambiguity, there will usually be two or more rows in the results, one for each possible meaning represented by an active concepts. However, in some cases it is possible that only one of the possible meanings is represented by an active concept.
</td>
</tr>
</table>

Footnotes

[1] Language and/or dialect should be specified if the service returns terms associated with referenced concepts.

[2] In the Snowstorm service requests [snowstorm] should be replaced by the URL to the Snowstorm server endpoint.

[3] In the FHIR service requests [fhir] should be replaced by the URL to the FHIR terminology server endpoint. FHIR® is a registered trademarks of HL7 (www.hl7.org).

[4] The SNOMED CT MySQL example database is not designed as a terminology server and is not intended for use in a live system . It is referenced in this guide as an illustration that some readers may find helpful. For more information about the SNOMED CT example database see the SNOMED CT - SQL Practical Guide. For instructions on how to build the example database refer to Appendix A: Building the SNOMED CT Example Database.

for example,

4.12 Get Mapping Data

Overview

Mapping is the process of converting data from one code system, classification, or terminology to another code system, classification, or terminology.

- The *mapping* process includes the preparation and maintenance of resources used for converting data.
- SNOMED CT *mapping* resources are distributed as mapping reference sets.

The SNOMED CT reference set design provides a common foundation on which mapping resources that meet different requirements can be built. The guidance in this section relates to the general approaches to accessing data in a mapping reference set. Table 4.12-1 provides reference to specific mapping reference set types and supporting documentation related to the use of those reference sets.

Table 4.12-1: Map Reference Sets and Documentation

Map type	**Description**	**Reference Set File Format**	**Additional Documentation**
Simple map	Supports one to one maps to or from another code system	5.2.9 Simple Map Reference Set	
ICD classification maps	Supports maps from SNOMED CT to ICD-10 classifications. Enables mapping to different codes based on rules and priorities. Also allows maps from a single code to combinations of codes.	5.2.10 Complex and Extended Map Reference Sets	ICD-10 Mapping Technical Guide
Maps including additional data to represent attribute-value combinations, correlation and source of original code	Supports maps where some codes in the other code system represent a combination of and attribute with a value (e.g. finding site appendix). Also supports maps where the correlation between source and target needs to be specified and where there are intellectual property reasons for recording that either the SNOMED CT or other code were the original representations of the mapped meaning.	5.2.14 Map Correlation and Origin Reference Set	
Maps to SNOMED CT expressions	Supports maps from another code systems where the use of a SNOMED CT expression may capture the meaning of that code more accurately than can be achieved with a single concept.	5.2.15 Code to Expression Reference Set	

Requirements and Options

Terminology services should be able to provide access to sets of maps based either on the SNOMED CT concept or a code from the other code system.

- The concept involved in the map is represented by the referencedComponentId. As noted in 4.10 Get Data from a Reference Set, access to reference set data using a combination of refsetId and referencedComponentId is a general requirement.
- The other code is generally in a column referred to as mapTarget[1]. Support for access to maps using a combination of refsetId and mapTarget is therefore also required.

The required services are listed in Table 4.12-2. Additional requirements should be considered for cases where the map refers to an expression rather than a single concept.

> **Note**
>
> The services described in this section are limited to accessing the relevant map data. In simple cases, this may be sufficient to enable mapping. However, in other cases the services will provide client applications with options to display to the user and/or processable rules to be tested against other data to determine the appropriate map.

Table 4.12-2: Services Required

Service Name and Status	Input	Output
Get maps based on combinations of refsetId and referencedComponentId REQUIRED	• Edition and version • Reference set identifier • Identifier of concept to be mapped to target scheme • Optional: Language/dialect[2]	All map data associated with maps in the specified reference set in which the referencedComponentId matches the specified concept identifier. Gets maps from SNOMED CT to another code system, if the map data supports mapping in that direction.
Get maps based on combinations of refsetId and mapTarget REQUIRED	• Edition and version • Reference set identifier • Other code to be mapped to SNOMED CT. • Optional: Language/dialect[2]	All map data associated with maps in the specified reference set in which the mapTarget matches the specified other code. Gets maps from another code system to SNOMED CT if the map data supports mapping in that direction.
Get maps based on combinations of refsetId, referencedComponentId and mapTarget OPTIONAL	• Edition and version • Reference set identifier • Identifier of concept • Other code • Optional: Language/dialect[2]	All map data associated with maps in the specified reference set in which the referencedComponentId matches the specified concept identifier and the mapTarget matches the other code. There are two situations in which this can be useful: • To test whether a map exists between a concept and a specific code in the other code system. • To access information about maps between a concept and a specific code in another code system.

Interdependencies

Required By

- Use Cases
  - 3.2.7 Mapping Data to or from or from Another Code System

Depends On

- 4.1 Select Edition and Version
- 4.10 Get Data from a Reference Set

Service Examples

✓ The Snowstorm and FHIR examples are presented in plain text and URL encoded versions. Always use the "Encoded URL" when testing the example service requests. The plain text version is included to aid readability but using this version in a service request may result in errors. These errors result from characters that have to be encoded as they are not permitted in a URL (see IETF RFC1738).

Table 4.12-3: Snowstorm API

Service Name	API Call[3]	Result

Get maps based on combinations of refsetId and referencedComponentId

```
GET [snowstorm]/[branchPath]/members?referenceSet=[refsetid]&referencedComponentId=[conceptId]&active=true
```

Example 1. ICD-O Map for 1338007 |Basal cell carcinoma (morphologic abnormality)|

```
GET [snowstorm]/MAIN/2020-01-31/members?referenceSet=446608001&referencedComponentId=1338007&active=true
```

Encoded URL

```
GET [snowstorm]/MAIN%2F2020-01-31/members?referenceSet=446608001&referencedComponentId=1338007&active=true
```

Example 2. ICD-10 Map for 74400008 |Appendicitis|

```
GET [snowstorm]/MAIN/2020-01-31/members?referenceSet=447562003&referencedComponentId=74400008&active=true
```

Encoded URL

```
GET [snowstorm]/MAIN%2F2020-01-31/members?referenceSet=447562003&referencedComponentId=74400008&active=true
```

Example 3. ICD-10 Map for 196607008 |Esophageal ulcer due to aspirin|

Returns a JSON representation of the data in active rows in the specified mapping reference set with a referencedComponentId that matches the specified conceptId.

The examples illustrate the following points [4]
Example 1. A concept that maps to a single ICD-O code in a simple map refset.
Example 2. A concept that maps to a single ICD-10 classification code in a complex/extended map refset.
Example 3. A concept that maps to a combination of two classification codes (there are two map groups and each contains a single map row with priority 1)
Example 4. A concept that has a single map group containing two rows. The row with priority 1 has a rule that requires the patient age to be less than or equal to 15 years. If that rule is not matched the row with map priority 2 applies as it contains the rule "OTHERWISE".
Example 5. A concept that, like example 2, contains two map groups. However, in the case it each of those maps also include specific human readable advice "... MAPPED FOLLOWING WHO GUIDANCE | POSSIBLE REQUIREMENT FOR AN EXTERNAL CAUSE CODE"
The data returned for each refset member of an extended map (e.g. as used in examples 2-5) includes the following mapping data as subproperties of the additionalFields property:

Property	Description
mapGroup	An integer grouping a set of maps. One map should be chosen from each group.
mapPriority	The order in which map within a group should be checked.
mapRule	A machine-readable rule, that determines if a map applies.
mapAdvice	Advice to assist manual selection map from a group.
mapTarget	The target code from the other coding scheme (if this map is chosen based on the rule or advice).
correlationId	|Representing the correlation between the concept and the target code.
mapCategoryId	Represents the map category that applies to this map.

```
GET [snowstorm]/MAIN/
2020-01-31/members?
referenceSet=447562003&refere
ncedComponentId=196607008&act
ive=true
```

Encoded URL

```
GET [snowstorm]/
MAIN%2F2020-01-31/members?
referenceSet=447562003&refere
ncedComponentId=196607008&act
ive=true
```

Example 4. ICD-10 Map for 32398004 |Bronchitis|

```
GET [snowstorm]/MAIN/
2020-01-31/members?
referenceSet=447562003&refere
ncedComponentId=32398004&acti
ve=true
```

Encoded URL

```
GET [snowstorm]/
MAIN%2F2020-01-31/members?
referenceSet=447562003&refere
ncedComponentId=32398004&acti
ve=true
```

Example 5. ICD-10 Map for 111613008 |Closed skull fracture with intracranial injury|

```
GET [snowstorm]/MAIN/
2020-01-31/members?
referenceSet=447562003&refere
ncedComponentId=111613008&act
ive=true
```

Get maps based on combinations of refsetId and mapTarget	`GET [snowstorm]/[branchPath]/members?referenceSet=[refsetid]&mapTarget=[otherCode]&active=true`
	Example 6. Map from ICD-O code "8090/3" to SNOMED CT concepts
	`GET [snowstorm]/MAIN/2020-01-31/members?referenceSet=446608001&mapTarget=8090/3&active=true`
	Encoded URL
	`GET [snowstorm]/MAIN%2F2020-01-31/members?referenceSet=446608001&mapTarget=8090%2F3&active=true`
	Example 7. Maps from NHS Clinical Terms to a SNOMED CT concept
	`GET [snowstorm]/MAIN/2020-01-31/members?referenceSet=900000000000497000&mapTarget=Xa9C4&active=true`
	Encoded URL
	`GET [snowstorm]/MAIN%2F2020-01-31/members?referenceSet=900000000000497000&mapTarget=Xa9C4&active=true`
		Returns a JSON representation of the data in active rows in the specified mapping reference set with a mapTarget that matches the specified otherCode. This returns the same data as noted above,
		Example 6. Reverses the map shown in example 1. In this case it returns records showing several concepts that map to the this ICD-O code. As these are simple map with no rules, this result indicates that all these concepts map to the same less specific ICD-O code.
		Example 7 . Maps from NHS CTV3 codes (previously used in the UK) to SNOMED CT concepts. In this case the map returns a single map from any specified CTV3 code to the closes matching SNOMED CT concept[5].
		Warning
		The ICD-10 map is a unidirectional map from SNOMED CT concept to ICD-10 codes. The mapGroup, mapPriority, mapRule, mapAdvice and mapCategoryId properties of complex and extended apply to the set of maps **from** a SNOMED CT concept to a mapTarget. These rules cannot be interpreted if maps are accessed for specified targetCode. Therefore, this service should not be used to access maps of this type[6].

Get maps based on combinations of refsetId, referencedComponentId and mapTarget	`GET [snowstorm]/[branchPath]/members?referenceSet=[refsetid]&referencedComponentId=[conceptId]&mapTarget=[otherCode]&active=true`
	Example 8. Finds any maps from 111613008 |Closed skull fracture with intracranial injury| to ICD-10 code S06.90
	`GET [snowstorm]/MAIN/2020-01-31/members?referenceSet=447562003&referencedComponentId=111613008&mapTarget=S06.90&active=true`
	Encoded URL
	`GET [snowstorm]/MAIN%2F2020-01-31/members?referenceSet=447562003&referencedComponentId=111613008&mapTarget=S06.90&active=true`
		Returns a JSON representation of the data in active rows in the specified mapping reference set with a referencedComponentId that matches the specified conceptId and a mapTarget that matches the specified code from the other code system.
		Example 8 . Could be used either or both of the following:
		a. Test if there is a map between the SNOMED CT concept "Closed skull fracture with intracranial injury" and the ICD-10 code S06.9 "Unspecified intracranial injury".
		b. Provide access to the associated mapping rule, advice and correlation data for any maps between these two codes.

Table 4.12-4: FHIR API

Service Name	API Call[7]	Result

<table>
<tr>
<td><b>Get maps based on combinations of refsetId and referencedComponentId</b></td>
<td>GET [fhir]/ConceptMap/$translate?code=[componentId] &system=http://snomed.info/sct&source=[sourceCodeSystem]?fhir_vs&target=[sourceCodeSystem]&url=[version]?fhir_cm=[refesetId]
Example 1. ICD-O Map for 1338007 |Basal cell carcinoma (morphologic abnormality)|
GET [fhir]/ConceptMap/$translate?code=1338007&system=http://snomed.info/sct &source=http://snomed.info/sct?fhir_vs &target=http://snomed.info/sct?fhir_vs &url=http://snomed.info/sct/900000000000207008/version/20200131?fhir_cm=446608001
<i>Encoded URL</i>
GET [fhir]/ConceptMap/$translate?code=1338007&system=http%3A%2F%2Fsnomed.info%2Fsct+&source=http%3A%2F%2Fsnomed.info%2Fsct%3Ffhir_vs+&target=http%3A%2F%2Fsnomed.info%2Fsct%3Ffhir_vs+&url=http%3A%2F%2Fsnomed.info%2Fsct%2F900000000000207008%2Fversion%2F20200131%3Ffhir_cm%3D446608001%0A
Example 2. ICD-10 Map for 74400008 |Appendicitis|
GET [fhir]/ConceptMap/$translate?code=74400008&system=http://snomed.info/sct&source=http://snomed.info/sct?fhir_vs &target=http://hl7.org/fhir/sid/icd-10&url=http://snomed.info/sct/900000000000207008/version/20200131?fhir_cm=447562003
<i>Encoded URL</i>
GET [fhir]/ConceptMap/$translate?code=74400008&system=http://snomed.info/sct&source=http://snomed.info/sct?fhir_vs &target=http://hl7.org/fhir/sid/icd-10&url=http://snomed.info/sct/900000000000207008/version/20200131?fhir_cm=447562003</td>
<td>Returns a JSON representation of data about each of the target components.
The data returned for each concept includes:
• <b>boolean:</b> True if the concept could be translated successfully. The value can only be true if at least one returned match
• <b>match:</b> Each match represents data for the map or associated target. Note that there may be multiple matches, where each element represents a mapTarget. For each mapTarget, following data is provided
• <b>system</b>: the codesystem of the mapTarget
• <b>code:</b> The identifier of the mapTarget
ⓘ <b>Request response</b>
Be aware that the FHIR TS API only provides information about the mapTarget. However, no additional map data represented in the map reference set is provided. This service is therefore sufficient to represent simple maps, but data required to interpret complex maps is not made available.
<b>Example 1</b>. A concept that maps to a single ICD-O code in a simple map refset.
<b>Example 2</b>. A concept that maps to a single ICD-10 classification code in a complex/extended map refset.
<b>Example 3</b>. A concept that maps to a combination of two classification codes (there are two map groups and each contains a single map row with priority 1)
<b>Example 4</b>. A concept that has a single map group containing two rows. For this map, a rule will determine which target should be chosen. Please note that the response does <b>not</b> show the rules as they are stated in the map reference set.</td>
</tr>
</table>

<table>
<tr><td></td><td>Example 3. ICD-10 Map for 196607008 |Esophageal ulcer due to aspirin|
GET [fhir]/ConceptMap/$translate?code=196607008&system=http://snomed.info/sct&source=http://snomed.info/sct?fhir_vs &target=http://hl7.org/fhir/sid/icd-10&url=http://snomed.info/sct/900000000000207008/version/20200131?fhir_cm=447562003
<i>Encoded URL</i>
GET [fhir]/ConceptMap/$translate?code=196607008&system=http%3A%2F%2Fsnomed.info%2Fsct&source=http%3A%2F%2Fsnomed.info%2Fsct%3Ffhir_vs+&target=http%3A%2F%2Fhl7.org%2Ffhir%2Fsid%2Ficd-10&url=http%3A%2F%2Fsnomed.info%2Fsct%2F900000000000207008%2Fversion%2F20200131%3Ffhir_cm%3D447562003
Example 4. ICD-10 Map for 32398004 |Bronchitis|
GET [fhir]/ConceptMap/$translate?code=32398004&system=http://snomed.info/sct&source=http://snomed.info/sct?fhir_vs &target=http://hl7.org/fhir/sid/icd-10&url=http://snomed.info/sct/900000000000207008/version/20200131?fhir_cm=447562003
<i>Encoded URL</i>
GET [fhir]/ConceptMap/$translate?code=32398004&system=http%3A%2F%2Fsnomed.info%2Fsct&source=http%3A%2F%2Fsnomed.info%2Fsct%3Ffhir_vs+&target=http%3A%2F%2Fhl7.org%2Ffhir%2Fsid%2Ficd-10&url=http%3A%2F%2Fsnomed.info%2Fsct%2F900000000000207008%2Fversion%2F20200131%3Ffhir_cm%3D447562003</td><td></td></tr>
<tr><td><b>Get maps based on combinations of refsetId and mapTarget</b></td><td>N/A</td><td>The FHIR TS API does not support retrieval of map data based on a specific mapTarget</td></tr>
</table>

Table 4.12-5: MySQL Example Database

Service Name	SQL Query [8]	Result

Get maps based on combinations of refsetId and referencedComponentId	SELECT * FROM snap_refset_extendedmap WHERE active=1 AND refsetId=[refsetId] AND referencedComponentId=[conceptId];
	Example 1. ICD-O Map for 1338007 |Basal cell carcinoma (morphologic abnormality)|
	SELECT * FROM snap_refset_simplemap WHERE active=1 AND refsetId=446608001 AND referencedComponentId=1338007;
	Example 2. ICD-10 Map for 74400008 |Appendicitis|
	SELECT * FROM snap_refset_extendedmap WHERE active=1 AND refsetId=447562003 AND referencedComponentId=74400008;
	Example 3. ICD-10 Map for 196607008 |Esophageal ulcer due to aspirin|
	SELECT * FROM snap_refset_extendedmap WHERE active=1 AND refsetId=447562003 AND referencedComponentId=196607008;
	Example 4. ICD-10 Map for 32398004 |Bronchitis|
	SELECT * FROM snap_refset_extendedmap WHERE active=1 AND refsetId=447562003 AND referencedComponentId=32398004;
	Example 5. ICD-10 Map for 111613008 |Closed skull fracture with intracranial injury|
		Returns rows of data from the specified mapping reference set with a referencedComponentId that matches the specified conceptId.
		For notes on the individual examples refer to the first row in Table 4.12-3.

	`SELECT * FROM snap_refset_extendedmap`
	`    WHERE active=1 AND refsetId=447562003`
	`    AND`
	`referencedComponentId=111613008;`
	Alternative approach using a stored procedure. This does not require the specific map reference set type to be identified.
	`-- Alternative that also shows the terms associated`
	`-- with identifiers in the map records`
	`CALL snap_members([refsetId], [conceptId]);`	

Get maps based on combinations of refsetId and mapTarget	`SELECT * FROM snap_refset_extendedmap`
	`    WHERE active=1 AND refsetId=[refsetId]`
	`    AND mapTarget=[otherCode];`
	Example 6. Map from ICD-O code "8090/3" to SNOMED CT concepts
	`SELECT * FROM snap_refset_extendedmap`
	`    WHERE active=1 AND refsetId=446608001`
	`    AND mapTarget="8090/3";`
	Example 7. Maps from NHS Clinical Terms to a SNOMED CT concept
	`SELECT * FROM snap_refset_extendedmap`
	`    WHERE active=1 AND refsetId=900000000000497000`
	`    AND mapTarget="Xa9C4";`
		Returns rows of data from the specified mapping reference set with a mapTarget that matches the specified code in the other code system.
		For notes on the individual examples refer to the second row in Table 4.12-3.

<table>
<tr><td><b>Get maps based on combinations of refsetId, referencedComponentId and mapTarget</b></td><td>SELECT * FROM snap_refset_extendedmap
WHERE active=1 AND refsetId=[refsetId]
AND referencedComponentId=[conceptId]
AND mapTarget=[otherCode];
Example 8. Finds any maps from 111613008 |Closed skull fracture with intracranial injury| to ICD-10 code S06.90
SELECT * FROM snap_refset_extendedmap
WHERE active=1 AND refsetId=447562003
AND referencedComponentId=111613008
AND mapTarget="S06.90";
Example 9. Finds any maps from 111613008 |Closed skull fracture with intracranial injury| to ICD-10 codes in the S06 chapter
SELECT * FROM snap_refset_extendedmap
WHERE active=1 AND refsetId=447562003
AND referencedComponentId=111613008
AND mapTarget like "S06.%";</td><td>Returns rows of data from the specified mapping reference set with a referencedComponentId that matches the specified conceptId and a mapTarget that matches the specified code in the other code system.
For notes on example 8 refer to the third row in Table 4.12-3.
Example 9. provides an additional option, which may be useful code systems such as ICD-10 in which part of the code represents represent chapters or subdivision. It allows a search for a map from a concept to any code in a specified chapter.</td></tr>
</table>

Footnotes

[1] The name mapTarget was applied to the other code column in early maps that were intended for unidirectional maps from SNOMED CT to other code systems or classifications. Subsequently, the column name mapTarget has been used for the other (non-SNOMED CT) code even in cases where the map is intended to support maps from the other code to a SNOMED CT concept. While this name is misleading in these cases, the column has not been renamed as a name change is likely to impact some implementations.

[2] Language and/or dialect should be specified if the service returns terms associated with referenced concepts.

[3] In the Snowstorm service requests [snowstorm] should be replaced by the URL to the Snowstorm server endpoint.

[4]

The notes on properties of ICD-10 maps on this page are only a brief summary. Please refer to the ICD-10 Mapping Technical Guide for more complete documentation and examples.

[5] The UK NHS provides alternative mapping resources from NHS CTV3 to SNOMED CT, which support maps that take account of different ways in which the same source code is used.

[6] Exceptional use cases exist for accessing complex or extended maps using the mapTarget when developing and quality assuming maps. For example, to identify codes in the other code system that are not the targets of any maps or that are the target for multiple maps from different concept.

[7] In the FHIR service requests [fhir] should be replaced by the URL to the FHIR terminology server endpoint. FHIR® is a registered trademarks of HL7 (www.hl7.org).

[8] The SNOMED CT MySQL example database is not designed as a terminology server and is not intended for use in a live system . It is referenced in this guide as an illustration that some readers may find helpful. For more information about the SNOMED CT example database see the SNOMED CT - SQL Practical Guide. For instructions on how to build the example database refer to Appendix A: Building the SNOMED CT Example Database.

4.13 Get Concept Model Rules

Overview

The SNOMED CT concept model is the set of rules that determines the permitted sets of relationships between particular types of concepts.

- The *concept model* specifies the attributes that can be applied to concepts in particular domains and the ranges of permitted values for each attribute. There are additional rules on the cardinality and grouping of particular types of relationships.

Concept model rules are represented in a machine readable form by the members of the following four reference set types:

- 5.1 MRCM Domain Reference Set
    - Each row in this reference set defines a concept model domain. Each domain is a set of concepts which the concept model permits to be defined or refined, using a particular set of attributes and ranges.
- 5.2 MRCM Attribute Domain Reference Set
    - Each row in this reference set specifies an concept model attribute that can be used to represent a characteristic of the meaning of a concept or the nature of a refinement.
        - An *attribute* is assigned a value (attribute value pair) when used in the definition of a concept or in a postcoordinated expression.
        - The *attributes* that can be used in *definitions* or *refinements* are represented by a concepts that are subtypes of the concept 410662002 |Concept model attribute (attribute)|.
        - The SNOMED CT concept model specifies:
            - The concept model domains which each specific *attribute* can be applied; and
            - The concept model range of values that can be applied to each specific *attribute*.
- 5.3 MRCM Attribute Range Reference Set
    - Each row in this reference set defines a concept model range that defines a set of values that the concept model permits to be applied to a specific attribute.
- 5.4 MRCM Module Scope Reference Set

- The members of this reference set specify the set of MRCM reference sets that should be applied to concepts a specified SNOMED CT module.

For full details of these reference set and the ways in which the rules are represented in these reference sets, refer to the SNOMED CT Machine Readable Concept Model (MRCM) specification and guide.

Requirements and Options

Access to concept model rules is required to support a range of different practical uses including those identified in the Interdependencies section on this page. The general techniques for accessing the data are the same as those for other references sets. However, interpretation of the data requires access to related data in each of the MRCM reference set types. To illustrate this the requirements listed in Table 4.13-1 and service examples are presented in a logical interrelated order.

Table 4.13-1: Services Required

Service Name and Status	Input	Output
Get the refsetId of specific type of MRCM reference set applicable to concepts in an identified module **REQUIRED INTERNAL SERVICE**	SNOMED CT Edition and Version The conceptId representing the moduleId The MRCM reference set type (mrcmDomain, mrcmAttributeDomain or mrcmAttributeRange)	The identifier of the specific MRCM reference set applicable to concept in the specified module. ⚠ Client applications do not require access to this data. However, it is required as an internal service to enable other MRCM services.

Service Name and Status	Input	Output
Get the domain or set of domains that include a specified concept **REQUIRED INTERNAL SERVICE**	SNOMED CT Edition and Version The conceptId for which the domains are to be identified. An indication of whether domains applicable to proximal primitive definitions are to be included. ⓘ The conceptId can be used to get the required moduleId and then, using the service above, the appropriate MRCM Domain reference set can be identified. The general process for identifying the domains that include a specified concept is as follows: • Identify the top-level domain that subsumes the concept. • Identify each of the domains that is a child or descendant of that top level domain. • The concept is included the top-level domain that subsumes the concept and any of its child and descendant domains that which have expression constraints to which it conforms. ▪ The expression constraint tested depends proximal primitive indicator value ◦ False: Conformance with the domainConstraint is required. ◦ True: Conformance with the proximalPrimitiveConstraint is required.	The identifiers of the domains to which the identified concept belongs. ⚠ Client applications do not require access to this data. However, it is required as an internal service to enable the service service following MRCM services.

Service Name and Status	Input	Output
Get the set of attribute rules applicable to an identified concept **REQUIRED**	SNOMED CT Edition and Version The conceptId for which the attribute rules are to be identified. The required internal services above are used to identify the domains that include the concept. An indication of whether domains applicable to proximal primitive definitions are to be included. Content type option - one of the following concept identifiers: 723593002 |All new precoordinated SNOMED CT content| 723594008 |All precoordinated SNOMED CT content| 723595009 |All postcoordinated SNOMED CT content| All the above content types subsume the content type 723596005 |All SNOMED CT content|. The content type 723593002 |All new precoordinated SNOMED CT content| also subsumes the content type 723594008 |All precoordinated SNOMED CT content|.	The selected set of attribute domain rules. - Each rule should include the content of all additional data columns in the MRCM Attribute Domain reference set. The superset of all attribute domain rules that apply to the identified domains apply to the identified concept. The set of rules returned should be filtered based on the selected content type.Note this is the superset that includes the attributes that can be applied to a top level domain and any sub domains of which a specified concept is a member.
Get the range of values applicable to a specified attribute **REQUIRED**	SNOMED CT Edition and Version The attributeId for which values are to be identified. Content type options (see previous service) The id of the concept to which the attribute is being applied (or the moduleId of the module to which concept belongs) are also required unless provided by internal services.	The selected set of attribute domain rules. - Each rule should include the content of all the additional data columns in the MRCM Attribute Range reference set. Conformance with the rangeConstraint expression is determined by applying the expression constraint to a specific value. The rangeConstraint can also be used filter the range of permitted refinements to the value of that attribute in postcoordinated expression.

Interdependencies

Required By

- Other Services
  - 4.14 Validate Concept Definitions and Expressions
  - 4.15 Test Expression Subsumption
- Use Cases
  - 3.2.3 EHR Data Entry
  - 3.2.2 EHR Data Entry Design
  - 3.4 EHR Reporting and Analytics
  - 3.8 Support Terminology Authoring and Review

Depends On

- 4.1 Select Edition and Version
- 4.2 Get a Concept, Description or Relationship
- 4.4 Get Definition of a Concept
- 4.5 Get and Test Concept Subtypes and Supertypes
- 4.6 Get and Test Reference Set Membership
- 4.7 Validate and Apply Expression Constraints
- 4.10 Get Data from a Reference Set

Service Examples

The Snowstorm and FHIR examples are presented in plain text and URL encoded versions. Always use the "Encoded URL" when testing the example service requests. The plain text version is included to aid readability but using this version in a service request may result in errors. These errors result from characters that have to be encoded as they are not permitted in a URL (see IETF RFC1738).

Table 4.13-2: Snowstorm API

Service Name	API Call[2]	Result

Get the set of reference sets applicable to concepts in an identified module	This service is supported internally and used by the services below. The relevant data is also directly accessible at the API using the general service used to access refset members as described in 4.10 Get Data from a Reference Set.
	`[snowstorm]/snomed-ct/[branchPath]/members?referenceSet=[MrcmModuleScopeRefsetId]&referencedComponentId=[moduleId]&active=true`
	For example
	`[snowstorm]/snomed-ct/MAIN/2020-01-31/members?referenceSet=723563008&referencedComponentId=900000000000207008&active=true`
	Encoded URL
	`[snowstorm]/snomed-ct/MAIN%2F2020-01-31/members?referenceSet=723563008&referencedComponentId=900000000000207008&active=true`
		JSON representation of three refset members each of which refers to one of the MRCM reference sets for concepts in the specified module.
		• The MRCM reference set references is represented by the additionalFields.mrcmRuleRefsetId property.

Get the domain or set of domains that include a specified concept	This service is supported internally and used by the services below. However, it is not directly accessible at the API.	

Get the set of attributes applicable to a concept in a specified domain or set of domains	`[snowstorm]/snomed-ct/mrcm/[branchPath]/domain-attributes?parentIds=427019001&proximalPrimitiveModeling=%5Btrue%7Cfalse%5D&contentType=%5Bcontent-type%5D%09`
	For example
	`[snowstorm]/snomed-ct/mrcm/MAIN/2020-01-31/domain-attributes?parentIds=427019001&proximalPrimitiveModeling=false&contentType=POSTCOORDINATED`
	Encoded URL
	`[snowstorm]/snomed-ct/mrcm/MAIN%2F2020-01-31/domain-attributes?parentIds=427019001&proximalPrimitiveModeling=false&contentType=POSTCOORDINATED`
		Returns a JSON representation of data related to each for the attributes applicable to specified concept.
		The data returned is filtered so it only includes data relevant to in the specified contentType.
		The data returned includes information from all the columns in the Attribute Domain reference set.
		• It also includes terms associated with the attribute.
		• The contentTypeId and ruleStrengthId are represented by human readable tokenized strings rather than the conceptId.

Get the range of values applicable to a specified attribute	Snowstorm does not provide a specific service for returning the attribute range constraint applicable to a concept model attribute. However, it does support access to the relevant data in two ways. Firstly the general service for access to refset members as (see 4.10 Get Data from a Reference Set) can be as illustrated here. Additionally, the Snowstorm service shown in the next row provides a way to carry out a search that is constrained to the permitted set of values for an attribute.
	`[snowstorm]/snomed-ct/[branchPath]/members?referenceSet=[MrcmAttributeRangeRefsetId]&referencedComponentId=[attributeId]&active=true&contentType=[contentType]`
	For example
	`[snowstorm]/snomed-ct/MAIN/2020-01-31/members?referenceSet=723562003&referencedComponentId=260870009&active=true&contentType=POSTCOORDINATED`
	Encoded URL
	`[snowstorm]/snomed-ct/MAIN%2F2020-01-31/members?referenceSet=723562003&referencedComponentId=260870009&active=true&contentType=POSTCOORDINATED`
	x
		Returns a JSON representation of data related to the specified attribute range.
		The range data is returned in the following properties of the additionalFields object:
		• rangeConstraint
		• attributeRule
		• contentTypeId
		• ruleStrengthId

Find values that are within the range or a specified attribute	`[snowstorm]/snomed-ct/mrcm/[branchPath]/attribute-values/260870009?contentType=POSTCOORDINATED&termPrefix=[searchTerm]`
	For example
	`[snowstorm]/snomed-ct/mrcm/MAIN/2020-01-31/attribute-values/260870009?contentType=POSTCOORDINATED&termPrefix=urg`
	Encoded URL
	`[snowstorm]/snomed-ct/mrcm/MAIN%2F2020-01-31/attribute-values/260870009?contentType=POSTCOORDINATED&termPrefix=urg`
		Returns a JSON representation of the search results constrained by the rangeConstraint.
		• The data returned is in the same form as documented in the 4.8 Find Concepts section.

Table 4.13-3: FHIR API

Service Name	API Call[3]	Result
Get concept model rules	N/A	No FHIR terminology services are available to provide detailed information about SNOMED CT concept model rules.

Table 4.13-4: MySQL Example Database

Service Name	SQL Query [4]	Result

Get the set of reference sets applicable to concepts in an identified module	`SELECT`
	`mrcmRuleRefsetId,refsetType`
	`FROM`
	`snap_refset_mrcmmodulescope m`
	`JOIN config_refsets r ON`
	`r.refsetId=m.mrcmRuleRefsetId`
	`WHERE active=1 AND`
	`refsetId=723563008`
	`AND`
	`referenceComponentId=[moduleId`
	`];`
	For example to get the MRCM refsetIds for 900000000000207008 |SNOMED CT core module|
	`SELECT`
	`mrcmRuleRefsetId,refsetType`
	`FROM`
	`snap_refset_mrcmmodulescope m`
	`JOIN config_refsets r ON`
	`r.refsetId=m.mrcmRuleRefsetId`
	`WHERE active=1 AND`
	`m.refsetId=723563008`
	`AND`
	`referencedComponentId=90000000`
	`0000207008;`
	Alternative approach to get refsetId for a specific MRCM refset for a specific module 900000000000207008 |SNOMED CT core module|
		The refsetIds and reference set types of the three MRCM references sets that apply to the specified module.

Get the refsetId of specific type of MRCM reference set applicable to concepts in an identified module	`SELECT getMrcmRefsetId([moduleId], [refsetType]);`
	The MRCM refsetType can be specified either by a single letter (D, A or R) or the full name (mrcm**D**omain, mrcm**A**ttributeDomain or mrcmAttribute**R**ange). The first example illustrates use of the full name while the others use the abbreviated style.
	`SELECT getMrcmRefsetId(900000000000207008,"mrcmDomain");`
	`SELECT getMrcmRefsetId(900000000000207008,"A");`
	`SELECT getMrcmRefsetId(900000000000207008,"R");`
		Each function call returns a single refsetId.
		The other MRCM services described below use this function to determine which reference set to use to access domain, attribute and range constraint information.

Get the domain or set of domains that include a specified concept	`CALL getMrcmDomains([conceptId],[proxPrim],@domainId,@domainList);`
	`-- Following line shows the results`
	`SELECT @domainId,@domainList;`
	Example 1. Get the domains that include the concept 71341001 |Bone structure of femur|
	`CALL getMrcmDomains(71341001,0,@domainId, @domainList);`
	`-- Following line shows the results`
	`SELECT @domainId, @domainList;`
	Example 2. Get the domains that include the concept 241075002 |Plain X-ray of femur|
	`CALL getMrcmDomains(241075002,0,@domainId, @domainList);`
	`-- Following line shows the results`
	`SELECT @domainId, @domainList;`
	Example 3. Get the proximal primitive domains that include the concept 241075002 |Plain X-ray of femur|
	`CALL getMrcmDomains(241075002,1,@domainId, @domainList);`
	`-- Following line shows the results`
	`SELECT @domainId, @domainList;`
		This procedure returns two values.
		• domainId - the most specific domain that includes the specified concept.
		• domainList - a comma separated list of all the domains that include the specified concept.
		Example 1 returns the following set of domains:
		• **domainId**: 723264001
		• **domainList**: 123037004,91723000, 723264001
		▪ 123037004 |Body structure|
		▪ 91723000 |Anatomical structure|
		▪ 723264001 |Lateralizable body structure reference set|
		Example 2 returns the following set of domains
		• **domainId**: 386053000
		• **domainList:** 71388002,386053000
		▪ 71388002 |Procedure|
		▪ 386053000 |Evaluation procedure |
		Example 3 returns the following set of proximal primitive domains
		• **domainList:** 71388002,386053000,387713003,433590000
		• 71388002 |Procedure|
		• 386053000 |Evaluation procedure|
		• 387713003 |Surgical procedure|
		• 433590000 |Administration of substance via specific route|
		ⓘ The procedure adds also adds a row to the config_resultsets table for each of the domains. This makes it easier for the services below to access to look up the attribute and range constraints for the set of domains applicable to a specified concept.

<table>
<tr>
<td><b>Get the set of attributes applicable to a concept in a specified domain or set of domains</b></td>
<td>
<pre><code>CALL
getMrcmAttributeRanges([concep
tId],[proxPrim],
[contentTypeKey],@ruleSetKey);
-- Following statement shows
the attribute ids and terms
SELECT DISTINCT
 `attributeId`,`term`
  FROM `config_mrcmrulesets`
`c`
  JOIN `snap_pref` `p` ON

`c`.`attributeId`=`p`.`concept
Id`
  WHERE
 `rulesetKey`=@rulesetKey;
</code></pre>
<b>Example 1</b>. Get the attributes applicable to the concept 71341001 |Bone structure of femur| when using postcoordination
<pre><code>CALL
getMrcmAttributeRanges(7134100
1,0,'pos',@ruleSetKey);
-- Following statement shows
the attribute ids and terms
SELECT DISTINCT
 `attributeId`,`term`
  FROM `config_mrcmrulesets`
`c`
  JOIN `snap_pref` `p` ON

`c`.`attributeId`=`p`.`concept
Id`
  WHERE
 `rulesetKey`=@rulesetKey;
</code></pre>
<b>Example 2</b>. Get the attributes and attribute constraints applicable to the concept 71341001 |Bone structure of femur| when using postcoordination.
<pre><code>CALL
getMrcmAttributeRanges(7134100
1,0,'pos',@ruleSetKey);
-- Following statement shows
the attribute ids and terms
SELECT
 `attributeId`,`grouped`,
`attributeCardinality`,
`attributeInGroupCardinality`,
`attributeContentTypeId`,
`attributeRuleStrengthId`,
`rangeConstraint`,
</code></pre>
</td>
<td>The getMrcmAttributeRanges procedure generates rows in the config_mrcmrulesets table. It returns a rulesetKey which can be used to query the generated rows. These rows contain not only the attributes but also range constraints.

Both examples get the attributes applicable to the same concept but the results returned by the select queries vary in content.

Example 1 returns the attributeId and preferred terms for each attribute applicable to the specified concept in the context specified by the proximal primitive and content type settings.

Example 2 returns all valid combinations of attributes and attribute range constraints applicable to the specified concept specified in the context specified by the proximal primitive and content type settings.</td>
</tr>
</table>

```
`rangeContentTypeId`,
`rangeRuleStrengthId`
  FROM `config_mrcmrulesets`
WHERE
 `rulesetKey`=@rulesetKey;
```

Get the range of values applicable to a specified attribute	As noted above, the getMrcmAttributeRanges procedure returns the attribute ranges of all attributes applicable to a concept. However, it is also possible to return the attribute range for specific attribute.
	SET @attributeId=[attributeId];
	SET @contentTypeIdList=[contentTypeIdList];
	SET @moduleId=[moduleId];
	SELECT `rangeConstraint`, `contentTypeId`, `ruleStrengthId`
	  FROM `snap_refset_mrcmattributerange`
	  WHERE `refsetId`=getMrcmRefsetId(@moduleId,"R");
	  AND `r`.`active`=1
	  AND FIND_IN_SET(`r`.`contentTypeId`,@contentTypeIdList)>0;
	Example. Get all the attribute range constraints applicable to the concept 71341001 |Bone structure of femur| when using postcoordination
	SET @attributeId=260870009;
	SET @contentTypeIdList="723596005,723594008";
	SET @moduleId=900000000000207008;
	SELECT `rangeConstraint`, `contentTypeId`, `ruleStrengthId`
	  FROM `snap_refset_mrcmattributerange`
	  WHERE `refsetId`=getMrcmRefsetId(@moduleId,"R")
	  AND `referencedComponentId`=@attributeId
	  AND `active`=1
		Returns the rangeConstraint for the attribute and the the contentTypeId and ruleStrengthId.
		ⓘ In some cases more than one row may be returned where there are rules with different strengths or applicable to different subsumed contentTypes.

	AND `FIND_IN_SET(`contentTypeId`,@contentTypeIdList)>0;`	

Footnotes

[1] Language and/or dialect should be specified if the service returns terms associated with referenced concepts.

[2] In the Snowstorm service requests [snowstorm] should be replaced by the URL to the Snowstorm server endpoint.

[3] In the FHIR service requests [fhir] should be replaced by the URL to the FHIR terminology server endpoint. FHIR® is a registered trademarks of HL7 (www.hl7.org).

[4] The SNOMED CT MySQL example database is not designed as a terminology server and is not intended for use in a live system . It is referenced in this guide as an illustration that some readers may find helpful. For more information about the SNOMED CT example database see the SNOMED CT - SQL Practical Guide. For instructions on how to build the example database refer to Appendix A: Building the SNOMED CT Example Database.

4.14 Validate Concept Definitions and Expressions

Overview

Concept Definitions

A concept definition is a set of one or more axioms that partially or sufficiently specify the meaning of a SNOMED CT concept.

Concept Definition Validation Requirements

Concept definitions must conform to the SNOMED CT Concept Model as specified by the machine readable concept model. A concept with a definition that does not conform to the *concept model* cannot be accurately classified. Therefore, it cannot be reliably tested for subsumption by another *concept* or inclusion in the results of an expression constraint or analytics query.

Service Requirements for Concept Definition Validation

- Terminology services that are designed to support SNOMED CT authoring must support validation of concept definitions based on concept model rules.
- Terminology services that are only designed to support use of the terminology do not need to meet this requirement.

SNOMED CT Expressions

An expression is a structured combination of one or more concept identifiers that represents an idea.

Example

- The *expression* 284196006 |:363698007=770850006| represents a burn to the skin of left index finger.
- An *expression* can also include the terms related to each of the identified concepts as shown below:

```
284196006 |burn of skin| : 363698007 |finding site| = 770850006 |Skin structure of left index finger|
```

Alternatives

Expression Validation Requirements

Expressions should conform to the SNOMED CT Concept Model as specified by the machine readable concept model. If an *expression* conforms to the SNOMED CT concept model, it can be classified by a description logic classifier in the same way as a *concept*. This means, it can be reliably tested for subsumption by other *concepts* and for inclusion in the results of applying an expression constraint or analytics query.

Expressions that do not conform to the SNOMED CT concept model cannot be accurately classified or tested against expression constraints. However, as noted in the next section, it may be possible to transform some non-conformant expressions so that they conform to the *concept model*.

Transforming Invalid Expressions to Valid Expressions

An expression may be invalid when tested against the concept model but may contain refinements that would be valid if correctly structured. There are several situations where predictable structural adjustments enable a valid expression to be constructed from an informal expression that does not fully confirm to the concept model.

These situations include:

- Moving a refining attribute to an attribute group in which it is valid.
- Moving a refining attribute so it provides a nested refinement to the value of another attribute (e.g. as illustrated by Example 4.14-1)
- Moving a context attribute (e.g. 408732007 |subject relationship context|, 408729009 |finding context|, 408730004 |procedure context| or 408731000 |temporal context|) to a context wrapper with the clinical focus concept or expression becoming the value of the 246090004 |associated finding| or 363589002 |associated procedure| attribute.

Example 4.14-1: Example Transformation to Apply Laterality Values to a Body Structures

A common situation in which it is useful to move a refining attribute is where a user has refined a disorder with the laterality attribute. The intention is clearly to state that the disorder is present in one limb or one side of the body. However, the concept model requires that the refinement is applied to the affected body structure not to the disorder. The laterality attribute should be applied as a nested refinement to the body structure specified by the 363698007 |finding site| in the concept definition or in the expression. If the body structure is lateralizable, this will result in a valid expression as illustrated below. The same approach can be applied to procedures with laterality refinements. However, in this case the laterality attribute should be applied as a nested refinement to the body structure specified by the 363704007 |procedure site| in the concept definition or in the expression.

```
 1. An user records "fracture of left femur" in a way that creates the following expression

71620000 |fracture of femur| : 272741003 |laterality| = 7771000 |left|

 2. The concept model rules indicate that this expression is invalid
 3. This is because concept model rules do NOT allow the "laterality" attribute to be applied to the co
ncepts in the "clinical finding" or "disorder" domains
```

4. As shown below the definition of "fracture of femur" includes "finding site" = "bone structure of femur"

```
=== 7523003 |Injury of thigh| +
   46866001 |Fracture of lower limb| :
       { 363698007 |Finding site| = 71341001 |Bone structure of femur| ,
         116676008 |Associated morphology| = 72704001 |Fracture| }
```

4. The concept model rules indicate the "bone structure of femur" can be refined by the "laterality" attribute
5. So the value "left" can be applied to this concept as shown below
6. The end result is an valid expression representing a fracture of the left femur which can be appropriately classified

```
=== 7523003 |Injury of thigh| +
   46866001 |Fracture of lower limb| :
       { 363698007 |Finding site| =( 71341001 |Bone structure of femur| : 272741003 |laterality| =
7771000 |left| ),
         116676008 |Associated morphology| = 72704001 |Fracture| }
```

Service Requirements for Expression Validation

- Terminology services that are designed to support use of postcoordinated expressions should support expression validation.

Requirements and Options

The required services are listed in Table 4.14-1.

Table 4.14-1: Services Required

Service Name and Status	Input	Output
Validate expression RECOMMENDED	- Edition and Version - Expression to be validated - Optional: Language/dialect[1]	- Indication of whether the expression is valid or invalid - If the expression is invalid an indication of the reason for failure of validation ▪ Syntax error including: ◦ Character position of first syntax error, and/or ◦ More specific error message(s). ▪ Concept model validation error including: ◦ Character position of start of invalid element and/or ◦ Reference to failed concept model rule(s). ▪ Terming errors including: ◦ Indication of any terms in the expression that do not match valid descriptions for the identified concept.
Transform invalid expression OPTIONAL	- Edition and Version - Invalid candidate expression for adjustment	- Structurally adjusted valid expression ▪ See Transforming Invalid Expressions to Valid Expressions - An error message if it is not possible to transform to valid expression
Validate concept definition(s) ONLY REQUIRED FOR AUTHORING	- Edition and Version - Identifier(s) of concept(s) to be validated	- Indication of validation results for one or more concepts including ▪ Nature of error ▪ If appropriate a reference to a failed concept model rule

Interdependencies

Required By

- Use Cases
  - 3.2.3 EHR Data Entry
  - 3.2.2 EHR Data Entry Design
  - 3.8 Support Terminology Authoring and Review

Depends On

- 4.1 Select Edition and Version
- 4.2 Get a Concept, Description or Relationship
- 4.3 Get Terms for a Concept
- 4.4 Get Definition of a Concept
- 4.5 Get and Test Concept Subtypes and Supertypes
- 4.8 Find Concepts
- 4.13 Get Concept Model Rules

Service Examples

> The Snowstorm and FHIR examples are presented in plain text and URL encoded versions. Always use the "Encoded URL" when testing the example service requests. The plain text version is included to aid readability but using this version in a service request may result in errors. These errors result from characters that have to be encoded as they are not permitted in a URL (see IETF RFC1738).

Table 4.14-2: Snowstorm API

Service Name	API Call[2]	Result
Validate expression	N/A	Not directly supported ⚠ SNOMED International does provide access to a tool that allows validation of the the syntax on an expression https://www.ihtsdo.org/SCT-cg.html.
Transform invalid expression	N/A	Not supported
Validate concept definition	`POST [snowstorm]/browser/{branch}/validate/concept` This service requires the concept to be posted as a JSON object representing the concept, its descriptions, axioms and relationships.	⚠ This command require the use of Drools (see https://www.drools.org/) and access to the Drools representation of the concept model rules.
Validate concept definitions	`POST [snowstorm]/browser/{branch}/validate/concepts` This service requires the concepts to be posted as a JSON array of objects representing the concept, its descriptions, axioms and relationships.	⚠ This command require the use of Drools (see https://www.drools.org/) and access to the Drools representation of the concept model rules.

Table 4.14-3: FHIR API

Service Name	**API Call**[3]	**Result**
Validate expression	N/A	No FHIR terminology services are available to validate SNOMED CT expressions.
Transform invalid expression		
Validate concept definition		
Validate concept definitions		

Table 4.14-4: MySQL Example Database

Service Name	**SQL Query** [3]	**Notes**
Validate expression	Not directly supported	The SQL example database does not provide direct support for expression validation. However, it does support access to concept model rules (see 4.13 Get Concept Model Rules). Using those services together with the service that applies expression constraints as described below enable a significant mount of expression validation. • The domains to which the focus concept belong can be checked using the **get domain or set of domains that include a specified concept** service. • The attributes in the expression can be checked to confirm they are applicable to those domains using the **get the set of attributes applicable to a concept in a specified domain or set of domains** service • The range constraints applicable to each attribute can be access using the **get the range of values applicable to a specified attribute** service and then applied to validate the values using the 4.7 Validate and Apply Expression Constraints service. • Where necessary these services can also be applied recursively to nested expressions representing the values of attributes.
Transform invalid expression	Not supported	
Validate concept definitions	Not required	The SQL example database is designed to provide read only access to the terminology. Therefore, it does not support requirements related to authoring or validating concept definitions.

Footnotes

[1]

Language and/or dialect should be specified if the service returns terms associated with referenced concepts.

[2] In the Snowstorm service requests [snowstorm] should be replaced by the URL to the Snowstorm server endpoint.

[3] The SNOMED CT MySQL example database is not designed as a terminology server and is not intended for use in a live system . It is referenced in this guide as an illustration that some readers may find helpful. For more information about the SNOMED CT example database see the SNOMED CT - SQL Practical Guide. For instructions on how to build the example database refer to Appendix A: Building the SNOMED CT Example Database.

4.15 Test Expression Subsumption

This page is a placeholder for an identified advanced service requirement for which a definitive set of detailed requirements are not yet available.

The detailed requirements are under discussion and this page will be updated based on the outcome of those discussions.

Overview

An expression is a structured combination of one or more concept identifiers that represents an idea. Expressions can be used to represent concept definitions. They can also be used to represent meanings that are not currently represented by a concept in the current SNOMED CT release.

When expressions are used, it is necessary to determine whether the meaning of a particular expression is a subtype of a specified concept or more generally is subsumed by a particular expression constraint. The terminology services required are similar to those described for testing concepts in sections 4.5 Get and Test Concept Subtypes and Supertypes and 4.7 Validate and Apply Expression Constraints. However, in this case the first step is to determine the meaning of the expression relative to existing defined concepts.

Requirements and Options

The required services are listed in Table 4.15-1.

Table 4.15-1: Services Required

Service Name and Status	Input	Output
Test subsumption of an expression minimal **REQUIRED**		

Interdependencies

Required By

- Use Cases
  - 3.2.3 EHR Data Entry (if entry of post-coordinated expressions is supported)
  - 3.4 EHR Reporting and Analytics (if stored data includes post-coordinated expressions)

Depends On

- 4.1 Select Edition and Version
- 4.4 Get Definition of a Concept
- 4.5 Get and Test Concept Subtypes and Supertypes
- 4.7 Validate and Apply Expression Constraints

Service Examples

✓ The Snowstorm and FHIR examples are presented in plain text and URL encoded versions. Always use the "Encoded URL" when testing the example service requests. The plain text version is included to aid readability but using this version in a service request may result in errors. These errors result from characters that have to be encoded as they are not permitted in a URL (see IETF RFC1738).

Table 4.15-2: Snowstorm API

Service Name	API Call[2]	Result
Get		
Get		

Table 4.15-3: FHIR API (*to be added*)

Service Name	API Call[3]	Result
...		
...		

Table 4.15-4: MySQL Example Database

Service Name	SQL Query [4]	Result
Get		

Footnotes

[1] Language and/or dialect should be specified if the service returns terms associated with referenced concepts.

[2] In the Snowstorm service requests [snowstorm] should be replaced by the URL to the Snowstorm server endpoint.

[3] In the FHIR service requests [fhir] should be replaced by the URL to the FHIR terminology server endpoint. FHIR® is a registered trademarks of HL7 (www.hl7.org).

[4] The SNOMED CT MySQL example database is not designed as a terminology server and is not intended for use in a live system . It is referenced in this guide as an illustration that some readers may find helpful. For more information about the SNOMED CT example database see the SNOMED CT - SQL Practical Guide. For instructions on how to build the example database refer to Appendix A: Building the SNOMED CT Example Database.

5 Service Implementation Considerations

SNOMED CT is made available to license holders in release packages containing sets of files that represent SNOMED CT components and. The content of these files provides the source data to which a terminology service provides access. This chapter provides guidance on how the structure and content of different release packages should be accessed and used to enable desired services. This chapter is important for those designing, developing, and maintaining terminology services, as they require a thorough understanding of the structure and content of different release packages and the release files they contain

Additional References

The section should be read in conjunction with more specific information in the following documents

- SNOMED CT Release File Specifications
- Extensions Practical Guide
- Reference Sets Practical Guide

5.1 Terminology Data Storage and Access

Requirements

- **Terminology service providers** should select data storage solutions that enable optimized access to SNOMED CT terminology data. They should also design service interfaces that facilitate access to the services identified in 4 Terminology Service Types.
- **Healthcare application providers** should evaluate the accessibility, performance and resource requirements of terminology services before committing to use of a particular solution.
- **Terminology service users** must ensure that the user experience when accessing terminology services directly or through chosen healthcare applications is acceptable in terms of ease of use and performance.

SNOMED CT terminology data is distributed in release packages containing tab-delimited release files that conform to the SNOMED CT Release Files Specification. All *release files* follow a consistent design pattern with the following key features:

- A versioning mechanism, which tracks additions, changes, and inactivations made in each release. This versioning mechanism enables terminology services to access the full history of every SNOMED CT component and reference set member.
- A limited range of data types, all of which are rendered as text in release files.
- Use of the Unicode UTF-8 character set for the string data type, to support the inclusion of the full range of Unicode characters.

Data from the files in SNOMED CT release packages needs to be loaded into a datastore that can be accessed by terminology services. The design of this datastore is a key design decision for developers of terminology services since it will determine the capabilities and performance characteristics of those services.

General requirements for the datastore design include:

- It must enable the representation of all the information contained in all release files types in the SNOMED CT release package(s) imported
- It must support the terminology services that are marked as *Required* in the subsections of 4 Terminology Service Types
- It must deliver scalable optimization of terminology services required by the use cases in 3 Terminology Service Use Cases that the server is designed to support[1].

Different technical options are available for implementing terminology services, for example, using a relational database[7], other database options (such as Graph databases [4],[5],[6]), or using predefined services accessible via an API (for example SNOMED International's Snowstorm[2]).

Footnotes

[1] A terminology service provider may choose to limit the range of use cases supported by their server. For example, terminology services required to enable 3.5 Reference Set Editing or 3.8 Support Terminology Authoring and Review may not be supported but some servers. Similarly, some terminology service providers may not optimize services that are used less frequently. For example, terminology services required to enable 3.7 Terminology Change Management are only required when intermittently when terminology update are being reviewed and applied.

[2] Snowstorm is a scalable SNOMED CT terminology server. It can be downloaded and installed from SNOMED International's GitHub site https://github.com/IHTSDO/snowstorm.

[3] Elasticsearch® is a registered trademark of Elasticsearch B.V. (see https://www.elastic.co/elasticsearch/).

[4] Neo4j®, is a registered trademark of Neo4j Inc (see https://neo4j.com/).

[5] A Neo4j upload script for SNOMED CT release files is available on GitHub (see https://github.com/IHTSDO/snomed-database-loader/tree/master/NEO4J). This script was kindly provided by Scott Campbell and his team from the University of Nebraska Medical Center, Omaha,

[6] Papers related to the use of Neo4j with SNOMED CT include: *An alternative database approach for management of SNOMED CT and improved patient data queries,* Scott Campbell et al, Journal of Biomedical Informatics https://www.researchgate.net/publication/281015927 and *Implementation of a Terminology Server with SNOMED CT in Graph Databases,* Elizabeth Silva Layes et al, MedInfo 2019 (see http://ebooks.iospress.nl/publication/52311).

[7] The SNOMED CT - SQL Practical Guide includes an example database in which tables are direct matched to the release file structures. The tables are then indexed to optimize searches and joins between the tables that support the delivery of services. The example database is **not** designed to be used to deliver operational terminology services. However, it provides a more direct illustration of the relationship between the content of the release files and the services outlined in 4 Terminology Service Types. The guide also includes a link to a downloadable script that creates a MySQL database and loads SNOMED CT release files into.

5.2 Enabling Access to Editions and Versions

Two key factors determine the ability of an instance of a terminology server to provide access to a particular SNOMED CT versioned edition. Firstly, the data that forms part of that *versioned edition* must be loaded into a data store that the terminology service is able to access. Secondly, unless the terminology server only has access to one versioned edition, the terminology server must allow the edition and version to be specified by the client application (see 4.1 Select Edition and Version).

Table 5.2-1 shows some of the different ways in which a terminology server may be designed and configured to enable access to a specific SNOMED CT edition or versioned edition Most of the options in Table 5.2-1 require the client application to select the edition and version to which they require access.

Table 5.2-1: Enabling Edition and Versioned Edition Access

Versioned Edition Accessibility	Description	Versioned Edition Selection	Notes
Fixed versioned edition	A specific instance of a terminology server accesses a datastore that only contains data for a single versioned edition.	Referring to a server instance implicitly identifies the edition and version.	**Not recommended** This option does not support the requirements specified in 3.7 Terminology Change Management. To meet those requirements, a terminology service must be able to access the version edition used by a client application prior to the most recent update.
Specific versions of a single edition	A specific instance of a terminology server accesses a datastore that only contains data for several specific versions of a single edition.	Referring to a server instance implicitly identifies the edition. The version can be selected from those available.	Recommended minimum for client applications only requiring access to a single edition. The accessible versions should include: • The current version(s) used by its client applications • The version(s) its client applications were using prior to their most recent updates.

Any versions of a single edition	A specific instance of a terminology server has access a datastore that contains the data required to access any version of a single edition. ✓ **Datastore options** Access to any version can be supported using a datastore designed in one of the following ways: 1. A datastore containing separate representations of every version of the edition. 2. A datastore containing data from the latest full release of the edition. Access to each required version is enabled using versioned views (see 5.3.2 Supporting Versioned Views). 3. A combination of the above options. Option 1 optimizes performance for access to the version(s) most frequently used by client applications. Option 2 enables access to other versions.	Referring to a server instance implicitly identifies the edition. Any version of that edition can be selected by specifying the effective time for a snapshot.	Recommended for client applications that are processing, reporting, or analyzing data collected using a range of different versions of a single edition.
Specific versions of several editions	A specific instance of a terminology server accesses a datastore that only contains data for several specific versions of a several editions.	The edition and version can be selected from those available.	Recommended for client applications that require access to multiple editions.

Any versions of several editions	A specific instance of a terminology server has access a datastore that contains data from the most recent full release for several editions. **Datastore options** Access to any version of several edition can be supported using a datastore designed in one of the following ways: 1. A datastore containing separate representations of every version of all supported editions. 2. A datastore containing separate representation of data from the latest full release of each edition. Access to each required version is enabled using versioned views. 3. A datastore containing data from the latest full release of all modules required by one or more of the supported editions. Access to a required versioned edition is enabled using versioned views of the modules that are part of the selected edition (see 5.3.2 Supporting Versioned Views).	The edition can be selected from those available. Any version of the selected edition can be selected by specifying the effective time for a snapshot.	Recommended for client applications that are processing, reporting or analyzing data collected with using a range of different versions of more than one edition.

	4. A combination of the above options. Option 1 optimizes performance for access to the versioned editions most frequently used by client applications. Option 2 or 3 enable access to other versioned editions.		

Table 5.2-2 summarizes the recommended ways to identify a SNOMED CT edition or versioned edition using a URI.

Table 5.2-2: Identifying SNOMED CT Editions and Versioned Editions

Method	Terminology Resource	General form and example	Notes
Module identifier and version date	**Edition**	**{moduleId}** Example: **900000000000207008**	The **{moduleId}** is an SCTID that identifies the edition. It does this by referring to the moduleId of the most dependent module in the edition. The other modules in the edition are specified by the module dependencies of that module.
	Version	**{versionDate}** Example **20200131**	The **{versionDate}** is the effectiveTime of the set the set of module dependency reference set rows for this a version of this edition. This is formally represented using the format *YYYYMMDD* (see Time data type).
Uniform Resource Identifiers	**Edition**	http://snomed.info/sct/ **{moduleId}** Example: http://snomed.info/sct/ **900000000000207008**	The SNOMED CT URI Standard defines globally unique identifiers for a wider range of terminology components. These URIs include the **{moduleId}** and **{versionDate}** noted above (for further details see URI Standard section 2.1 URIs for Editions and Versions). ⚠ **Note** The URI http://snomed.info/sct/731000124108/version/20200301 refers to the 2020-01-31 version of the US Edition, which is formally defined to include the 731000124108 |US National Library of Medicine maintained module| and the two International Edition modules on which it depends (900000000000207008 |SNOMED CT core module| and 900000000000012004 |SNOMED CT model component module|).
	Versioned Edition	http://snomed.info/sct/ **{moduleId}**/version/ **{versionDate}** Examples: http://snomed.info/sct/ **900000000000207008**/version/ **20200131** http://snomed.info/sct/ **731000124108**/version/ **20200301**	

Other Options	**Edition**	Key, path or alias for an edition Example: {server-url}/MAIN/SNOMEDCT-US	A less formal approach that provides a more meaningful representation of the edition and version can be used. The example shown here is currently used by the Snowstorm terminology server to refer to the US Edition of SNOMED CT. ⚠ **Note** This Snowstorm path refers to an extended version of the formally defined US edition. In addition to the formally defined content noted above, it also includes the 5991000124107 |SNOMED CT to ICD-10-CM rule-based mapping module| and several other additional modules that are distributed by SNOMED International in separate release packages. The inclusion of these additional extension modules creates an extended versioned edition (see 5.2.2 Enabling Access to Extended Editions).
	Versioned Edition	Combination key, path or alias with the version date Example: {server-url}/MAIN/SNOMEDCT-US/2020-03-01	

5.3 Release Types and Versioned Views

5.3.1 Importing Release Types

Requirements

- **Organizations that maintain** SNOMED CT edition(s) **or** extension(s**)** must distribute one or more release package(s). Each *release package* must contain a full release for the current version of an *edition* or *extension* for which they are responsible. Optionally, each *release package* may also contain a snapshot release and/or delta release for the same version of the same *edition* or *extension*.
- **Terminology service providers** must ensure their service is able to import data from either a *full* or *snapshot release* of a selected *edition* or *extension*. They should also consider enabling the service to update a previously imported *edition* or *extension* using a *delta release*.
- **Healthcare application providers** should ensure that their application can work with terminology services that facilitate importing and updating SNOMED CT using the *release types* available for all the *SNOMED CT editions* and *extensions* to which their customers require access.
- **Terminology service users** must confirm that the healthcare applications and terminology services they procure can provide access to the *SNOMED CT editions* and *extensions* they need to access. They must also confirm that the terminology service facilitates updates to more recent versions of those *editions* and *extensions.*

Release Type Characteristics and Uses

Table 5.3.1-1 specifies the content of each of the release types and the types of use that these release types support.

Table 5.3.1-1: SNOMED CT Release Types

Release Type	Description	Usage notes
Full	A release type in which the release files contain every version of every component and reference set member ever released.	The set of *full release files* for a SNOMED CT edition provides data that enables terminology services to offer access to snapshot views of terminology content at any specified date between its first release and the current release. The data in a set of full release files for an edition also enables terminology services to provide information about changes made to the content of that edition between any two release dates.
Snapshot	A release type in which the release files contain only the most recent version of every component and reference set member released, as at the release date. The version of each component and reference set member in a snapshot release is the most recent version of that component at the time of the snapshot. • Components that have been added, changed or inactivated in the current release have an effectiveTime equal to the release date. • Components that have not changed since an earlier release have an effectiveTime that matches the release in which they were most recently added, changed or inactivated. • Components that are inactive are included and must be distinguished from active components based on the value in the active column.	A set of *snapshot release files* for a SNOMED CT edition provides data that enables terminology services to offer access to the current release snapshot view of the content of that edition. Access to snapshot release file data for two or more releases of an edition, also enables terminology services that provide information about changes made to that edition between the release dates of those snapshots.

Release Type	Description	Usage notes
Delta	A release type in which the release files contain only rows that represent component versions and reference set member versions created since the previous release date. Notes • Each *row* in a *delta release* file represents either a new component or reference set member, or a change to an existing component or reference set member since the previous release date. • A *delta release* identifies differences between two versions of the same release package. • A *delta release* added to the previous full release is identical to the full release of the new version. • The previous *release date*, on which a *delta release* is based, is usually the date of the most recent previous release. However, that may not always be the case. For example, where interim releases are made between two major releases there may be a combined *delta release* covering a period since a previous major release.	The content of a set of *delta release files* for a SNOMED CT edition cannot be used as a stand alone resource but can be used in the following ways: • To update the previous full release data: by adding the data in each delta release file to the current stored version of the full release. • To update the previous snapshot release data: by adding new components and replacing existing components with updated versions of those components. • To enable review of changes to components since the previous release: by comparing the delta data with the previous snapshot data for the same *edition*.

Comparing the Content of Different Release Types

Table 5.3.1-2 provides an illustration of the content of the delta, full and snapshot release types across five releases of a simplified example release file. The names of the four columns in this example file match four of the columns in the SNOMED CT descriptions release file. However, to simplify the example, the moduleId, conceptId and caseSignificanceId columns have been omitted, and the values in the id columns are simple integers rather than SCTIDs. Each of the five releases in the diagram is shown with a valid effectiveTime (for presentational clarity the release dates shown are a year apart and the year is shown in bold text). Table 5.3.1-3 provides a key to use of color shading to highlight additions, changes and inactivations in each version of the release.

Table 5.3.1-2: Illustration of Content of Different Release Types

Release and Notes	**Delta**	**Full**	**Snapshot**

20010131 - **The first release illustrating the initial content of the release file.**

- In this example, six components are created and included in the first release.

The delta, full, and snapshot release files are identical in the first release because there are no earlier versions of any components.

- The delta represents the changes from an empty starting file and thus, like the full release, it contains all the component versions ever created.
- Since there is only one version of each component, this is by definition the latest version of that component, so all the and as there is only one version of each identified component so the same data is also present in the snapshot file.

id	effectiveTime	active	term
1	**2001**0131	1	first
2	**2001**0131	1	second
3	**2001**0131	1	3rd
4	**2001**0131	1	forth
5	**2001**0131	1	V
6	**2001**0131	1	VI

id	effectiveTime	active	term
1	**2001**0131	1	first
2	**2001**0131	1	second
3	**2001**0131	1	3rd
4	**2001**0131	1	forth
5	**2001**0131	1	V
6	**2001**0131	1	VI

id	effectiveTime	active	term
1	**2001**0131	1	first
2	**2001**0131	1	second
3	**2001**0131	1	3rd
4	**2001**0131	1	forth
5	**2001**0131	1	V
6	**2001**0131	1	VI

20020131 - **The second release illustrating a change.**

- Component 3 term changed from "3rd" to "third".

The delta file only includes the changed item.

The full file includes all rows in the previous release plus the changed row with the new release date and updated term.

The snapshot view includes only the most recent version of each component. So the initial version of component 3 is omitted.

id	effectiveTime	active	term
3	**2002**0131	1	third

id	effectiveTime	active	term
1	**2001**0131	1	first
2	**2001**0131	1	second
3	**2001**0131	1	3rd
3	**2002**0131	1	third
4	**2001**0131	1	forth
5	**2001**0131	1	V
6	**2001**0131	1	VI

id	effectiveTime	active	term
1	**2001**0131	1	first
2	**2001**0131	1	second
3	**2002**0131	1	third
4	**2001**0131	1	forth
5	**2001**0131	1	V
6	**2001**0131	1	VI

20030131 - **The third release illustrating two additions.**

- Components 7 and 8 added with term "fourth" and "fifth" respectively.

The delta file only includes the added items.

The full file includes all rows in the previous release plus the added rows with the new release date.

The snapshot view includes only the most recent version of each component including the newly added components.

id	effectiveTime	active	term
7	**2003**0131	1	fourth
8	**2003**0131	1	fifth

id	effectiveTime	active	term
1	**2001**0131	1	first
2	**2001**0131	1	second
3	**2001**0131	1	3rd
3	**2002**0131	1	third
4	**2001**0131	1	forth
5	**2001**0131	1	V
6	**2001**0131	1	VI
7	**2003**0131	1	fourth
8	**2003**0131	1	fifth

id	effectiveTime	active	term
1	**2001**0131	1	first
2	**2001**0131	1	second
3	**2002**0131	1	third
4	**2001**0131	1	forth
5	**2001**0131	1	V
6	**2001**0131	1	VI
7	**2003**0131	1	fourth
8	**2003**0131	1	fifth

20040131 - **The fourth release illustrating inactivation.**

- Component 4 is inactivated.

The delta file only includes the inactivated item.

The full file includes all rows in the previous release and the inactivated row with the new release date.

The snapshot view includes only the most recent version of each component. So the only version component 4 included is the new inactive version.

id	effectiveTime	active	term
4	**2004**0131	0	forth

id	effectiveTime	active	term
1	**2001**0131	1	first
2	**2001**0131	1	second
3	**2001**0131	1	3rd
3	**2002**0131	1	third
4	**2001**0131	1	forth
4	**2004**0131	0	forth
5	**2001**0131	1	V
6	**2001**0131	1	VI
7	**2003**0131	1	fourth
8	**2003**0131	1	fifth

id	effectiveTime	active	term
1	**2001**0131	1	first
2	**2001**0131	1	second
3	**2002**0131	1	third
4	**2004**0131	0	forth
5	**2001**0131	1	V
6	**2001**0131	1	VI
7	**2003**0131	1	fourth
8	**2003**0131	1	fifth

<table>
<tr>
<td><b>2005</b>0131 - <b>The fifth release illustrating a typical situation with updates of different types.</b>
<ul>
<li>Component 5 is inactivated.</li>
<li>Component 6 is changed with the revised term "sixth".</li>
<li>Component 9 is added with the term "seventh".</li>
</ul>
</td>
<td>
<table>
<tr><th>id</th><th>effectiveTime</th><th>active</th><th>term</th></tr>
<tr><td>5</td><td><b>2005</b>0131</td><td>0</td><td>V</td></tr>
<tr><td>6</td><td><b>2005</b>0131</td><td>1</td><td>sixth</td></tr>
<tr><td>9</td><td><b>2005</b>0131</td><td>1</td><td>seventh</td></tr>
</table>
</td>
<td>
<table>
<tr><th>id</th><th>effectiveTime</th><th>active</th><th>term</th></tr>
<tr><td>1</td><td><b>2001</b>0131</td><td>1</td><td>first</td></tr>
<tr><td>2</td><td><b>2001</b>0131</td><td>1</td><td>second</td></tr>
<tr><td>3</td><td><b>2001</b>0131</td><td>1</td><td>3rd</td></tr>
<tr><td>3</td><td><b>2002</b>0131</td><td>1</td><td>third</td></tr>
<tr><td>4</td><td><b>2001</b>0131</td><td>1</td><td>forth</td></tr>
<tr><td>4</td><td><b>2004</b>0131</td><td>0</td><td>forth</td></tr>
<tr><td>5</td><td><b>2001</b>0131</td><td>1</td><td>V</td></tr>
<tr><td>5</td><td><b>2005</b>0131</td><td>0</td><td>V</td></tr>
<tr><td>6</td><td><b>2001</b>0131</td><td>1</td><td>VI</td></tr>
<tr><td>6</td><td><b>2005</b>0131</td><td>1</td><td>sixth</td></tr>
<tr><td>7</td><td><b>2003</b>0131</td><td>1</td><td>fourth</td></tr>
<tr><td>8</td><td><b>2003</b>0131</td><td>1</td><td>fifth</td></tr>
<tr><td>9</td><td><b>2005</b>0131</td><td>1</td><td>seventh</td></tr>
</table>
</td>
<td>
<table>
<tr><th>id</th><th>effectiveTime</th><th>active</th><th>term</th></tr>
<tr><td>1</td><td><b>2001</b>0131</td><td>1</td><td>first</td></tr>
<tr><td>2</td><td><b>2001</b>0131</td><td>1</td><td>second</td></tr>
<tr><td>3</td><td><b>2002</b>0131</td><td>1</td><td>third</td></tr>
<tr><td>4</td><td><b>2004</b>0131</td><td>0</td><td>forth</td></tr>
<tr><td>5</td><td><b>2005</b>0131</td><td>0</td><td>V</td></tr>
<tr><td>6</td><td><b>2005</b>0131</td><td>1</td><td>sixth</td></tr>
<tr><td>7</td><td><b>2003</b>0131</td><td>1</td><td>fourth</td></tr>
<tr><td>8</td><td><b>2003</b>0131</td><td>1</td><td>fifth</td></tr>
<tr><td>9</td><td><b>2005</b>0131</td><td>1</td><td>seventh</td></tr>
</table>
</td>
</tr>
</table>

Table 5.3.1-3: Key to Color Shading in the Release Type Illustration Table

Color	Description
White	Component unchanged in this release and this is the most recent version of this component.
Grey	This is not the most recent version of this component. One or more of the releases since the effectiveTime of this row have either changed or inactivated this component. As these rows are not the most recent version of the identified component, they do not appear in the snapshot view.
Green	The first version this component as added in this release.
Yellow	An updated version of an existing component. The updated version is present in the current release and includes a change to one or more data values[1].
Red	An updated version of an existing component that inactivates the component.

Footnotes

[1] The formal specification of these release types is in Section 3.2 Release Types of the SNOMED CT Release File Specifications.

[2] If the release package does not include a snapshot release or delta release, versioned views that match the content of these *release type*s can be derived from the full release.

5.3.2 Supporting Versioned Views

Requirements

- **All SNOMED CT release packages** must include a set of full release files for a SNOMED CT edition or extension. A *full release* includes the complete history of the content of that *edition* or *extension*. This allows a terminology server to enable access to versioned views of the content of this *edition* or *extension* as it was at any point between its first release and the current release. However, the extent to which a terminology service supports access to different *versioned views* may vary.
- **Terminology service providers** must ensure that their services optimize the performance of access to the current snapshot view of a selected *edition* or *extension*. They should also provide services that enable access to earlier *snapshot views* and delta views to support the analysis of data entered using earlier versions of the same *edition* or *extension* and terminology change management when updating to a newer version.
- **Healthcare application providers** must ensure that their applications can use terminology services that provide access to the versioned views of SNOMED CT required by their users. Services that support access to the *snapshot view* of the *versioned edition* currently in use should be selected by default and should be optimized. Services that enable access to a new *version* of the same *edition* should also be accessible to meet user requirements for managing updates to a newer *versioned edition*[1]. Access to earlier *versions* of the same *edition* should also be accessible to meet requirements for analysis of existing data recorded using terminology content that has subsequently been inactivated.
- **Terminology service users** must procure applications and terminology services that optimize access to the *snapshot view* of a selected versioned edition that they are currently using. They must also ensure that the applications and terminology services that they procure meet their requirements for managing updates to a newer *versioned edition*[1]. They should also ensure that the application and terminology services can meet their requirement for analysis of previously entered data recorded using terminology content that has subsequently been inactivated.

Types of Versioned Views

A versioned view is a set of component versions and reference set member versions defined by characteristics of their effectiveTimes.

- *Versioned views* and release types are closely related. A *release type* is a physical representation of a particular *versioned view*.
- Some *versioned* views are not instantiated as *release types* but all valid *versioned views* of a SNOMED CT edition can be generated from a full release of that *edition.*
- A delta view is a view of SNOMED CT that contains only rows that represent changes to components and reference set members since a specified date or between two specified dates in the past.
- A full view is a view of SNOMED CT that includes all versions of all components and reference set members in a full release.

- A snapshot view is a view of SNOMED CT that includes the most recent version of all components and reference set members at a specified point in time.

There are three types of versioned views:

- A snapshot view is a view of SNOMED CT that includes the most recent version of all components and reference set members at a specified point in time.
    - The current *snapshot view* is most important versioned view. It will be in constant use during day to day use of the terminology.
    - When updating to a new version, access to the previous earlier *snapshot view* is required to determine the type of change made to each component and reference set member (see 4.9 Identify Changes to the Terminology).
    - When reviewing, analyzing or reporting data collected while using earlier versions, access to *snapshot views* of the terminology at the time of data collection may assist interpretation of any unexpected variations.
- A delta view is a view of SNOMED CT that contains only rows that represent changes to components and reference set members since a specified date or between two specified dates in the past.
    - When updating to a new version, a *delta view* identifies components and reference set members that were added, updated or inactivated between to releases. to terminology content (see 4.9 Identify Changes to the Terminology).
- A full view is a view of SNOMED CT that includes all versions of all components and reference set members in a full release.
    - The ability to access the current full view may be useful for an organization that does not apply every update of the edition that they use. For example, if they receive data from another organization that includes content that is not in the snapshot view they are using.
    - In exceptional circumstances, access to earlier versions of the full view may also be useful for detailed forensic review of data and resolution of anomalies.

Terminology Service Versioned View Options

Table 5.3.2-1 identifies options for the *snapshot views* that a SNOMED CT terminology server may support and Table 5.3.2-2 identifies the types of delta views that may be supported.

Table 5.3.2-1: Snapshot views that may be supported by terminology servers

View	Description

Single *snapshot view* of an edition	Some terminology services may only support access to a single *snapshot view* of a specified SNOMED CT edition at a one preset point in time. • The specified time in this case is the current version in use by the client application. Individual service requests cannot specify a different version. • When a decision is made to update to a newer version the server would be reconfigured to change the specified versioned view. In most cases, a terminology service that supports a single *snapshot view* of a specified edition only needs to import the relevant snapshot release files for that edition[2]. A single *snapshot view* of an edition: • Meets requirements for current data entry and display. ▪ This includes the display of data entered using earlier versions of the selected edition. Even if a concepts is now inactive, it will still have at least one active description. • Does not meet requirements for change management when assessing and updating to a more recent version of the terminology. • Does not meet the requirements for analysis of retrospective data that may have been recorded and previously analysed when an earlier version of the terminology was in use. • Does not meet requirements for access to terminology data in SNOMED CT modules that are not part of the specified SNOMED CT edition.
Multiple *snapshot views* of an edition	Some terminology services may support access to *snapshot views* of a specified SNOMED CT edition for a limited number of specified points in time. • The specified times in this case would typically include the current version in use by the client application and the version in use prior to the most recent update. Individual service requests can specify any of the supported versions. • When a decision is made to update to a newer version the server would be reconfigured to add the new versioned view to the set of supported views. A terminology service that supports multiple versioned views of a specified edition can do this in one of two ways: • Importing the snapshot release for each version of the edition that needs to be accessed[2]; or • Importing the full release for the most recent version of the edition and then optimizing access to the required snapshot views[3]. Access to multiple *snapshot views* of an edition: • Meets requirements for current data entry and display. ▪ This includes the display of data entered using earlier versions of the selected edition. Even if a concept if now inactive, it will still have at least one active description. • Meets requirements for change management when assessing and updating to a more recent version of the terminology. • Partially meets requirements for analysis of retrospective data that may have been recorded and previously analysed when an earlier version of the terminology was in use. ▪ This is limited to data that was originally recorded using one of the accessible versioned views. • Does **not** meet requirements for access to terminology data in SNOMED CT modules that are not part of the specified SNOMED CT edition.

All *snapshot views* of an edition	Some terminology services may support access to *snapshot views* of a specified SNOMED CT edition at any specified specified point in time. • The required version is specified by the client application as part of each terminology service call. A terminology service that supports all *snapshot views* of a specified edition can do this in one of three ways: • Importing the snapshot release for all versions of the edition[2]; or • Importing the full release for the most recent version of the edition and then optimizing access to all snapshot views; or • Importing the full release for the most recent version of the edition to enable access to all snapshot views and specifically optimizing access to the current *snapshot view* and any other versioned that is likely to be frequently accessed[3]. A terminology service that supports all *snapshot views* of a specified edition needs to import the full release files for that edition. Access to all *snapshot views* of an edition: • Meets requirements for data entry and display. • Meets requirements for change management when assessing and updating to a more recent version of the terminology. ▪ Full change management functionality requires services that support comparisons of a component between versions. • Meets requirements for analysis of retrospective data that may have been recorded and previously analysed when an earlier version of the terminology was in use. • Does **not** meet requirements for access to terminology data in SNOMED CT modules that are not part of the specified SNOMED CT edition.
All *snapshot views* of a range of different editions [4]	Some terminology services may support access to *snapshot views* of a range of different SNOMED CT editions at any specified specified point in time. • The required edition and version is specified by the client application as part of each terminology service call. A terminology service that supports a range of *snapshot views* of a specified edition needs to import the full release files for all the modules required by one or more of the supported editions. Access to all *snapshot views* of a range of different editions: • Meets requirements for data entry and display. • Meets requirements for change management when assessing and updating to a more recent version of the terminology. ▪ Full change management functionality requires services that support comparisons of a component between versions. • Meets requirements for analysis of retrospective data that may have been recorded and previously analysed when an earlier version of the terminology was in use. • Meets requirements for access to terminology data in SNOMED CT modules that are part of any of the supported SNOMED CT editions.

Table 5.3.2-2: Delta views that may be supported by terminology servers

View	**Description**

Single *delta view* of changes between current version and previous version of an edition	Some terminology services may only support a single *delta view* of changes between current version and previous version of an edition. This is useful for reviewing the most recent changes to the terminology. However, a complete review of the impact of those changes also requires access the snapshot view of the previous release, as this is the baseline for the changes shown in the delta. (see 4.9 Identify Changes to the Terminology).
Multiple *delta views* of changes between a selected version and the preceding version of the same edition	Multiple *delta views* of changes between a specified version and the preceding version of the same edition. This is useful for reviewing the changes between earlier releases. It provides an opportunity to retrospectively remedy issues arising from change. However, complete review of the impact of those changes also requires access the snapshot view of the release prior to each delta view, as this is the baseline for the changes shown in the delta. (see 4.9 Identify Changes to the Terminology).
Configurable *delta views* of changes to an *edition*	Configurable *delta views* of changes to an edition between any two specified points provide a flexible solution that can identify a sequence of changes occurring to one or more components. Fully configurable *delta views* require access to data for a *full release* of the *edition*. This also provides the authoritative source for the starting point for all the changes shown in the delta.
Configurable *delta views* of a range of available *editions*	Configurable *delta views* of changes to one of a number of editions between any two specified points in time provide additional value in cases where more than one edition is in use.

Creating and Updating Versioned Views

Creating and Updating Full Views

To enable terminology services to access a full view of a SNOMED CT edition, a server must be able to import a full release of that edition into an appropriate datastore.

To enable access to a more recent full view of a SNOMED CT edition, a server may:

- Import the latest full release as a replacement for the previous full release data; or
- Import the latest delta release and append this data to the previous full release data.

Creating and Updating Snapshot Views

To provide access to the current snapshot view of a SNOMED CT edition, a server may:

- Import the current snapshot release for that edition;
- Selectively import data from the current full release for that edition, including only the most recent versions of each component or reference set member up to and including a specified snapshot time; or
- Enable access to the full view of that edition to be filtered to include only the most recent version of each component prior to a specified snapshot time;

To provide access to more recent snapshot views of a SNOMED CT edition, a server may:

- Use any of the above methods for providing access to a *snapshot view* for a different date, either in addition to or as a replacement for a previously imported snapshot view; or
- Import the latest delta release and use this to update the previous *snapshot view* by appending new components and replacing existing components with updated versions.

Creating and Updating Delta Views

To provide access to the delta view for the current version of a SNOMED CT edition, a server may:

- Import the delta release for that edition into an appropriate datastore;
- Selectively import data from the current full release for that edition, including only component versions with effectiveTime values that are after the previous release date and are also on or before the current release date;

- Enable access to the full view of that edition to be filtered to include only versions of components with effectiveTime values that are after the previous release date and on or before the current release date.

To provide access to other delta views of a SNOMED CT edition, a server may:

- Use any of the above methods to create delta views between other release dates. In all cases the effectiveTime must be after one release date and on or before the next release date;
- Import the full release and provide delta views by selectively filtering data to ensure the effectiveTime is greater than a specified start date and less than or equal to a specified end date.
    - This method can be used to generate configurable delta views between specified dates that may not be directly related to adjacent release dates. In this type of delta view, there may be more than one version of a component with the specified date range. In this case, it may be useful to consider two possible delta views. One including only the most recent change to each component in the specified period and the other including all changes during that period.

Footnotes

[1] For more information about version access requirements to support version update see 3.7 Terminology Change Management.

[2] Organizations responsible fornational editions and extensions are only required to distribute a full release of their content. Therefore, terminology services that only support access to a specified set of snapshot views may need to enable importing of those snapshots from a full release. However, in practice most organizations also distribute a snapshot releases so this requirement may only occur in exceptional circumstances.

[3] The snapshot release files for the 2020-01-31 International Edition requires 1.25 Gb of storage while the full release files requires just under 2 Gb of storage. Therefore, services that provide access to two or more separate snapshot views are likely to require more storage than services that use the full release to provide access to a complete range of snapshot views. However, storing and indexing the data for individual snapshots is likely to deliver significant performance advantages as it avoids the need to generate the snapshot in response to each service request.

[4] It is theoretically possible for a terminology service to support any number of different SNOMED CT editions. However, as a large number of organizations are licensed to create SNOMED CT extensions, it is unlikely that a single instance of any terminology service would be required to provide access to every SNOMED CT edition.

5.4 Importing Release File Data

5.4.1 Checking Release Files

Requirements

- **Terminology service providers** must provide services that check the files in a release package for data integrity and internal consistency prior to importing data into a live terminology server. They must also ensure files in extension modules are checked for integrity and compatibility with the installed edition prior to importing the files into a live server.
- **Healthcare application providers** must ensure that their applications perform data integrity and compatibility tests prior to importing release packages and extensions modules.
- **Terminology service users** should procure and use healthcare applications that use terminology services that perform data integrity, compatibility and compatibility tests prior to importing release packages and extensions modules.

Quality Assurance Requirements

It is important to check the integrity of a SNOMED CT release package before importing it into an operational terminology server for the first time. The extent and depth of quality assurance checking should take account of past experience of the quality of release packages received from the responsible organization. SNOMED International, applies a robust quality assurance process to the International Edition and those responsible for established National Editions follow similar processes. Nevertheless, it is recommended that the General Release Package Integrity Checks described below are carried out on all release packages. In the case of extension release packages, it is strongly recommended that the Additional Checks for Extension Release Files are also carried out before importing data.

General Release Package Integrity Checks

To validate the integrity of SNOMED CT release packages, checks can be performed to confirm that:

1. The distribution files imported are all part of the same release:
2. The set of files imported is complete and includes all mandatory components.
3. In the case of a delta release, the data previously imported is the version immediately prior to the delta release being imported.
4. In the case of a snapshot or full release, pre-existing data has been removed:
    - Alternatively, the import process may be configured to overwrite duplicate rows so that:
        - The end result of a snapshot import does not contain any obsolete rows;
        - The end result of a full release import is identical to the content of the full release.
5. All component Identifiers have:
    - A partition identifier appropriate to the type of component;
    - A valid check-digit.
6. All fields meet data type, size and value constraints
    - These criteria are specified in the following sections of the SNOMED CT Release File Specifications: 4 Component Release Files Specification and 5 Reference Set Release Files Specification.
7. All concepts must have at least two active descriptions (one of type 900000000000013009 |Synonym| and one of type 900000000000003001 |Fully specified name|).
8. All active concepts (except the root concept 138875005 |SNOMED CT Concept|) must have at least one active relationship of type 116680003 |Is a|.

Other consistency checks may also be applied to ensure the integrity of the data.

Additional Checks for Extension Release Files

The process of importing an extension is similar to importing the main distribution files. However, some additional checks are required to ensure appropriate installation, maintenance, and use of an extension.

Applications should check each extension prior to installation to ensure that:

1. It is an extensions recognized and formally approved for use by a suitable authority within the user organization.
2. It is supported by, or based on, the currently installed International edition version.
3. The required versions of other extensions on which this extension depends have already been installed (or have been selected for installation as part of the same import process).
    - All dependencies of the extension module are represented by rows in the Module Dependency Reference Set.
        - The relevant rows have a moduleId value that matches the moduleId of the extension and a sourceEffectiveTime that matches the current version of the extension.
        - Each of these rows represents a dependency on a module version represented by the referencedComponentId and targetEffectiveTime values.
4. The installation procedure has pre-checked all components in the extension to ensure that:
    - All components have a moduleId that is valid for the extension.
    - All component identifiers:

    - Are unique for a particular effectiveTime;
    - Have a partition identifier appropriate to the type of component;
    - Have a namespace Identifier appropriate to the provider of that extension;
    - Have a valid check-digit;
- All fields meet data type, size and value constraints specified for the relevant tables.

Warning

If any components fail any of these tests the entire extension should be rejected. Rejecting individual components is liable to lead to inconsistent data. Accepting data that fails these test may create conflicts between different extensions or between an extension and the International or National edition.

5.4.2 Importing Release Files

Requirements

- **Terminology service providers** must provide services that enable access to all the data in a selected versioned view of a selected SNOMED CT edition. This requirement applies to all *editions* and *versioned views* that the terminology services are able to access. The data must be stored in forms that enable it to serve the purposes identified in the SNOMED CT Release Files Specifications. However, the data storage format is not required to precisely match the data structure in the release files.
- **Healthcare application providers** must ensure that their applications support the use of terminology services that allow access to all the data in selected *versioned views* of a selected *SNOMED CT edition*. They should ensure that their applications are able to use this data to meet user requirements in ways that make effective use of SNOMED CT features and guidance [1].
- **Terminology service users** should procure and use healthcare applications and terminology services that enable them to realize the benefits of SNOMED CT by accessing all the data in a selected *edition* and making effective use of terminology design features and guidance [1].

A terminology service must provide access to all the data in one or more specified *versioned views* of a *selected edition*. The simplest way to achieve this is to import the content of all the files in the full release and/or the snapshot release from the release package representing the selected edition. However, where this is not possible Table 5.4.2-1, identifies alternative sources that can be used.

Table 5.4.2-1: Alternative Sources for Importing Release Views

Exception	Notes
Importing snapshot release data from a full release of the same versioned edition.	Selectively process rows in each release files in the following way. • From each file, only import rows with the most recent *effectiveTime* for each unique *id*. • Other rows must **not** be imported.
Importing snapshot release data from a full release of a later version of the same edition.	Selectively process rows in each release files in the following way. • Discard or ignore all rows with an *effectiveTime* that is more recent than the required version date. • From each file, only import the retained rows with the most recent *effectiveTime* for each unique *id*. • Other rows must **not** be imported.

Importing delta release data from a full or snapshot release of the same versioned edition.	Selectively process rows in each release files in the following way. • Only import rows with an *effectiveTime* greater than the previous release date. • Other rows must **not** be imported.
Importing delta release data of changes between any two versions from a full release of the same versioned edition or a later version of the same edition.	Selectively process rows in each release files in the following way. • Only import rows that have an *effectiveTime* that is ▪ Greater than the starting version date; and ▪ Less than or equal to the ending version; and ▪ If more than one row for a unique *id* meets the above criteria, only import the row that has the most recent *effectiveTime*. • Other rows must **not** be imported.
Importing edition data from a release package containing more than one edition	Identify the module dependencies of the edition to be imported. Selectively process rows in each release file in the following way: • Only include rows that matches either the edition *moduleId* or the *moduleId* of a module on which the edition depends.
Importing edition data from multiple release packages	Identify the module dependencies of the edition to be imported. Selectively process rows in each release file in the following way: • Only include rows that matches either the edition *moduleId* or the *moduleId* of a module on which the edition depends. If the initial package does not contain all the necessary versioned modules to satisfy the module dependencies, import additional data from release files in other packages. • Avoid importing the same component version from more than one package (i.e. skip duplicates where the *id* and *effectiveTime* both match).

The data structure of each release file is specified the following sections of the SNOMED CT Release File Specifications 4 Component Release Files Specification and 5 Reference Set Release Files Specification.

Appendix A: Summary of Types and Use Cases

Table Appendix A:-1 provides a summary of the main types of terminology service requirements. For each requirement type, use cases are identified with a more specific description of the terminology services required to address that use case.

Table Appendix A:-1: Terminology service requirement types summary

Required Terminology Services	Use Cases	Application of Terminology Services to Specific Use Cases
4.1 Select Edition and Version	All	Unless the server only hosts one edition and version these settings are required prior to or as part of service request.
4.2 Get a Concept, Description, or Relationship	3.1 Explore and Review SNOMED CT	Get specified concepts to enable exploration of their descriptions, definitions and position in the hierarchy.
	3.2.3 EHR Data Entry	Record data in an EHR using a single concept.
	3.2.2 EHR Data Entry Design	Create or edit a user interface template or control that records specific concepts in an EHR depending on user selections.
	3.4 EHR Reporting and Analytics	Add selected concepts to constrain or extend EHR query criteria.
	3.5 Reference Set Editing	Create or edit subsets, maps (or other artifacts represented by SNOMED CT reference sets).
	3.8 Support Terminology Authoring and Review	Select supertypes, attributes, and values to define a concept.
4.3 Get Terms for a Concept	3.1 Explore and Review SNOMED CT	Display terms for concepts in search results, hierarchy views, and concept definitions.
	3.2.2 EHR Data Entry Design	Display terms for a concept and enable the selection of a term for display in a user interface template or control.
	3.2.3 EHR Data Entry	Display terms in user interface template or control.
	3.3 Display EHR Data	Display terms for concepts recorded in an EHR.
	3.4 EHR Reporting and Analytics	Display terms for concepts to provide human-readable EHR reports and analytics results.
	3.8 Support Terminology Authoring and Review	Display terms for concepts in search results, definitions, expressions, and expression constraints.
4.4 Get Definition of a Concept	3.1 Explore and Review SNOMED CT	Display inferred necessary normal form definition of a concept from relationships.
	3.1 Explore and Review SNOMED CT	Display the stated definition of a concept from stated OWL axioms.
	3.8 Support Terminology Authoring and Review	Display stated and inferred necessary normal form definitions of a concept in the authoring environment.
4.5 Get and Test Concept Subtypes and Supertypes	3.1 Explore and Review SNOMED CT	Display subtype and supertypes in a hierarchical view.
	3.2.2 EHR Data Entry Design	Enable selection of subtype constraints on data entry template controls.
	3.2.3 EHR Data Entry	Apply subsumption tests to constrain concept searches and selection for data entry.
	3.4 EHR Reporting and Analytics	Apply subsumption tests in reporting and analytics queries.
4.6 Get and Test Reference Set Membership	3.2.2 EHR Data Entry Design	Enable selection of reference set membership constraints on data entry template controls.

	3.2.3 EHR Data Entry	Apply reference set membership test to constrain concept searches and selection for data entry. Display members of a reference set as options for data entry.
	3.4 EHR Reporting and Analytics	Apply reference set membership tests in reporting and analytics queries.
	3.5 Reference Set Editing	List the members of a reference set.
4.7 Validate and Apply Expression Constraints	3.2.2 EHR Data Entry Design	Create or edit a user interface template including expression constraints that limit permitted values that can be entered through a specific data entry control.
	3.2.3 EHR Data Entry	Apply expression constraints to concept searches for data entry.
	3.4 EHR Reporting and Analytics	Create or edit reporting specifications using expressions constraints to represent query criteria Apply expression constraints in reporting and analytics queries.
	3.5 Reference Set Editing	Create or edit subsets using expression constraints to intensionally define their members.
4.8 Find Concepts	3.1 Explore and Review SNOMED CT	Find concepts to be displayed.
	3.2.3 EHR Data Entry	Find concepts to be entered.
	3.2.2 EHR Data Entry Design	Find concepts for inclusion in a template.
	3.4 EHR Reporting and Analytics	Find concepts to be used as inclusion criteria in reports and analytic queries.
	3.8 Support Terminology Authoring and Review	Find concepts to be used in the definition of another concept.
4.9 Identify Changes to the Terminology	3.7 Terminology Change Management	Identify components that have been added, changed or inactivated since the previous release.
	3.7.3 Manage Impact of Changes on EHR Applications	Review user interface templates and controls for concepts or descriptions that are now inactive. Review existing EHR records for concepts that are now inactive. Review existing EHR queries for inclusion (by reference or subsumption) of concepts and that are now inactive.
	3.7.4 Manage Impact of Changes on Extensions	Review extension for any dependencies on concepts and descriptions that have been inactivated in modules on which the extension depends.
4.10 Get Data from a Reference Set	3.1 Explore and Review SNOMED CT	Get acceptability of a specified description in an identified language reference set
4.11 Get History Data	3.7 Terminology Change Management	Identify reasons for inactivation of concepts used in existing EHR records and assess the impact of these records using these concepts on reports: • Get inactivation reason for a specified inactive concept or description • Get active concept(s) with historical associations from a specified inactive concept Produce human-readable reports of changes in a new release: • For each inactive concept, the report should include the reason for inactivation and potential replacements based on historical associations • For each inactive description, the report should include the reason for inactivation

	3.7.3 Manage Impact of Changes on EHR Applications	Use historical associations to identify potential alternative concepts to replace inactive concepts in: • Data entry templates • Queries used for reporting or analysis
	3.7.4 Manage Impact of Changes on Extensions	Use historical associations to identify potential alternative concepts to replace inactive concepts in definitions of extension concepts.
4.12 Get Mapping Data	3.2.7 Mapping Data to or from or from Another Code System	Get maps for a specified concept in an identified map reference set.
4.13 Get Concept Model Rules	3.2.3 EHR Data Entry	Facilitate, constrain, and validate refinements applied to entered concepts. This applies to refinement specifically entered by a user and to refinements created by natural language processing of free text. It also applies whether the refined data is represented in the record by a postcoordinated expression or by records that contain additional fields for specific refining values (e.g. laterality, body site, etc.).
	3.2.2 EHR Data Entry Design	Facilitate, constrain, and validate the design of data entry templates to ensure the use of the resulting template results in data that is consistent with the concept model.
	3.4 EHR Reporting and Analytics	Facilitate, constrain, and validate the creation of expression constraints and/or queries used to generate reports and to analyze records. Assuming the data complies with the concept model, queries should also conform to that model. However, queries may also be written to look for or apply reporting techniques to data that does not conform to the model.
	3.8 Support Terminology Authoring and Review	Facilitate, constrain, and validate the creation and modification of concept definitions.
4.14 Validate Concept Definitions and Expressions	3.2.3 EHR Data Entry	Record data in EHR using an expression (in cases where no specific concept matches the required meaning).
	3.2.2 EHR Data Entry Design	Create or edit user interface templates or controls that record specific expressions in an EHR depending user selections (in cases where no specific concept matches the required meaning).
4.15 Test Expression Subsumption	3.2.3 EHR Data Entry	Apply expression subsumption tests in an expression constraint or query.
	3.4 EHR Reporting and Analytics	Include subsumed or equivalent expressions in EHR data queries.

ⓘ Please note that the primary focus of this guide is on read-only services without a user-interface.

The rationale for focusing on **read-only** services is as follows:

- Read-only terminology services are applicable to all healthcare applications that require access to SNOMED CT.
- Add-update services are only required by those responsible for the development and maintenance of SNOMED CT editions or extensions. This smaller audience reduces the value of including detailed documentation of development services in this guide.
- Add-update services have a high degree of interdependence to ensure that updates are handled in ways that enable effective version control, validation, and quality-assured management and distribution. Therefore, describing generalized versions of individual update services without the surrounding architectural context would serve little purpose and could be misleading.
- The functionality of specific SNOMED CT authoring tools is described in the documentation associated with those tools. Given the limited number of organizations requiring these services, there is no clear case for replicating that material in this guide.

The rationale for focusing on services that **without a user interface** is as follows:

- Terminology services without a user interface can provide applications with access to the content of SNOMED CT.
- Terminology services without a user interface can provide access in ways that exploit the design features of the terminology while leaving each client applications to present data inline with its own design styles.
- Terminology services that include a user interface are in practice combinations of several general terminology services with one or more user interface components. There are many possible combinations of terminology services with specific user interface controls and it not feasible to document all of these. Instead, this guide notes a few general examples of ways in which terminology services described in this guide may be bound to user interface controls.

Appendix B: Editions, Versions and Extensions

Requirements

- **Terminology service providers** must deliver services that can load data from SNOMED CT release packages that conform to the SNOMED CT Release File Specifications. They must enable client applications to access the content of a selected SNOMED CT versioned editions. They should also enable access to extended versioned editions including additional modules that are compatible with the selected *versioned edition*. They must also provide access to data that identifies the *versioned edition* or *extended versioned edition* currently being accessed.
- **Terminology service users** must ensure that they procure and use terminology services that can be configured to enable access to the *SNOMED CT versioned editions* (or *extended versioned editions*) required by their organization, speciality or national authority.
- **Healthcare application providers** must ensure that their applications support access to terminology services that are configurable to enable access to the *SNOMED CT versioned editions* (or *extended versioned editions*) required by their customers. They must also ensure that their applications track the history of the *versioned editions* (*or extended versioned editions*) used to record and access data as this may affect interpretation and analysis of data.

A SNOMED CT edition is a complete set of SNOMED CT components and reference set members that either belong to an identified SNOMED CT module or belong to one of the modules on which that *module* depends.

- The module used to define the scope of an *edition* is referred as the focus module of that *edition*.
- All *SNOMED CT editions* (except the International Edition) are a combination of one or more extension modules, together with the modules from the SNOMED CT International Edition.
- A complete *SNOMED CT edition* may be prepared and released by SNOMED International or by the provider of a SNOMED CT Extension. Alternatively, a *SNOMED CT edition* may be derived from one or more release packages, by combining the contents of an identified *focus module* with the contents of the relevant version of all modules on which the *focus module* depends.
- The dependencies between modules are represented using the |Module dependency reference set (foundation metadata concept)|.
- A SNOMED CT edition can be identified using a Uniform Resource Identifier (URI) as specified by the URI Standard (2.1 URIs for Editions and Versions).

Table Appendix B:-1 lists the definitions of different types of *SNOMED CT editions* including formally defined *editions*, versions of *editions* and *editions* that have been extended by the addition of selected modules. Terminology services should enable client applications to access the content of selected *versions* of selected *editions* with or without selected additional *modules*.

Table Appendix B:-1: Edition Types

Edition Type	Definition	Notes

International Edition	The set of SNOMED CT components and reference set members that either belong to a specific module identified by SNOMED International as the focus module for that *edition* or belong to one of the *modules* on which that module depends.	• The *International Edition* includes the foundational content of SNOMED CT on which all other SNOMED CT modules must have dependencies. • SNOMED International currently identifies the |SNOMED CT core module (core metadata concept)| as the *focus module* for the *International Edition*. Only the |SNOMED CT model component module| is currently specified as a dependency. • The *International Edition* may be supplemented by extensions, maintained and distributed by Members and Affiliates, to meet additional national, local, and organizational requirements.
Member / National Edition	A set of SNOMED CT components and reference set members that either belong to a focus module identified by a National Release Center (NRC), or belong to one of the modules on which that *module* depends.	• The focus module is part of the National Release for which that NRC is responsible. • An NRC may have multiple *National Editions* with different focus modules for each edition. • A *National Edition* may: ▪ be part of a National Release distributed to licensees. ▪ combine a focus module from the *National Release*, the relevant versions of modules in the International Edition, and any other extension modules on which the focus module of the *National Edition* depends.
Affiliate Edition	A set of SNOMED CT components and reference set members that either belong to a focus module identified by an Affiliate Licensee with an allocated extension namespace identifier, or belong to one of the modules on which that *module* depends.	• The *focus module* is part of the extension for which that Affiliate is responsible. • An *Affiliate Edition* may extend the SNOMED CT International Edition or a National Edition.

Extended Edition[1]	A SNOMED CT edition to which a specified set of additional modules has been added.	• An *extended edition* provides a way to add a set of modules containing maps, subsets and other derivatives to an edition. • The module dependencies of all the additional *modules* must be satisfied by other *modules* in the *extended edition*. • Additional *modules* included in an *extended versioned edition* may include additional reference sets and reference set members and metadata concepts (with descriptions and a subtype relationship linking them to the appropriate branch of the |SNOMED CT Model Component| hierarchy). However, they should **not** include other concepts, descriptions, relationships or OWL reference set members.
Versioned Edition	A SNOMED CT edition that is published on a specific date.	• A new *version* of the International Edition of SNOMED CT is released twice a year (in January and July). • National extensions generally follow this cycle, however it is often with a three-month delay. Some extensions (notably those including medication-related concepts) are released more frequently.
Extended Versioned Edition	A versioned edition to which specified versions of additional modules have been added.	• An *extended versioned edition* provides a way to add a set of modules containing maps, subsets and other derivatives to a versioned edition. • An *extended versioned edition* is only valid if the module dependencies of all the additional *modules* are satisfied by other *modules* in that *extended versioned edition*. • Additional *modules* included in an *extended versioned edition* may include additional reference sets and reference set members and metadata concepts (with descriptions and a subtype relationship linking them to the appropriate branch of the |SNOMED CT Model Component| hierarchy). However, they should **not** include other concepts, descriptions, relationships or OWL reference set members.

Footnotes

[1] For more information about SNOMED CT extensions see the Extensions Practical Guide.