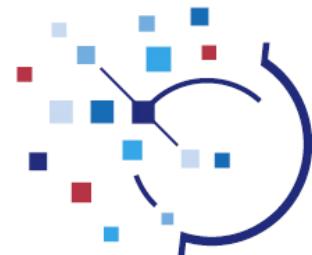


■ INTERNATIONAL HEALTH TERMINOLOGY
STANDARDS DEVELOPMENT ORGANISATION



SNOMED CT® Technical Implementation Guide

July 2013 International Release

(US English)

This document may have been updated since you received it.
The latest versions of IHTSDO documents are available online.
Latest PDF version: www.snomed.org/tig.pdf
Latest web browsable version: www.snomed.org/tig
Directory of available documents: www.snomed.org/doc

Contents

1 Preface.....	8
1.1 Notation used in this document.....	8
1.2 Document Properties.....	9
1.3 Amendment History	9
1.4 Status.....	12
1.5 Referencing and Commenting.....	13
1.6 Additional information.....	14
1.7 Inventory of Documentation.....	15
1.8 Where is the Glossary?.....	16
1.9 Copyright Notice.....	16
2 Overview of the guide.....	18
2.1 Who Should Read This Guide?.....	19
2.1.1 Software designers and developers.....	19
2.1.2 Health informatics specialists, analysts, purchasers and integrators.....	20
2.2 Important Notices.....	20
2.3 Additional information and feedback.....	21
3 SNOMED CT implementation	22
3.1 Motivation for Implementation.....	22
3.1.1 Benefits for electronic health records	23
3.1.2 Benefits for knowledge representation.....	23
3.1.3 Benefits of an open global approach.....	24
3.1.4 Benefits of extensibility and configurability.....	24
3.2 Implementation Types.....	25
3.2.1 Implementation Types - Clinical records.....	26
3.2.2 Implementation Types - Knowledge representation.....	27
3.2.3 Implementation Types - Aggregation and analysis.....	27
3.2.4 Terminology tools.....	28
3.3 Implementation Levels.....	29
3.3.1 Implementation Level - Scope of use.....	30
3.3.2 Implementation Level - Record structure.....	32
3.3.3 Implementation Level - Expression storage.....	33
3.3.4 Implementation Level - Data entry.....	33
3.3.5 Implementation Level - Data retrieval.....	34
3.3.6 Implementation Level - Communication.....	35
3.4 Implementation Services.....	36
3.4.1 Service architecture.....	37

4 Structure and Content Guide.....	40
4.1 SNOMED CT Technical Overview.....	40
4.1.1 Components.....	40
4.1.2 Derivatives.....	48
4.1.3 Extensions	51
4.1.4 Instance data.....	53
4.2 Logical Abstract Models.....	54
4.2.1 Logical Model of SNOMED CT Components	55
4.2.2 Logical Model of SNOMED CT expressions	72
4.3 Representational Forms.....	88
4.3.1 Release Files	88
4.3.2 Representing SNOMED CT identifiers	88
4.3.3 Representing Extensions	95
4.3.4 Representational Forms for Expressions	102
4.3.5 Stated Relationships Guide.....	110
4.3.6 Other Representational Forms.....	115
4.3.7 Additional Reference Materials.....	116
5 Release File Specifications.....	128
5.1 Release File Formats.....	128
5.2 SNOMED CT Editions, Extensions, Releases and Modules.....	129
5.3 Release Format 2 - Introduction.....	131
5.4 SNOMED CT - File Naming Conventions.....	131
5.4.1 File Naming Convention - Overview.....	132
5.4.2 FileType element.....	132
5.4.3 ContentType element.....	134
5.4.4 ContentSubType element.....	135
5.4.5 Country Namespace element.....	137
5.4.6 VersionDate element.....	137
5.4.7 Extension element.....	137
5.5 Release Format 2 - Core Component Guide.....	138
5.5.1 General.....	138
5.5.2 Relationships between files.....	143
5.5.3 File formats.....	144
5.5.4 Extensibility Mechanism.....	151
5.5.5 Metadata hierarchy.....	152
5.6 Release Format 2- Reference Sets Guide.....	155
5.6.1 Introduction.....	155
5.6.2 Reference Set Specifications.....	156
6 Concept Model Guide.....	191
6.1 Essential Features of the Concept Model	191

6.1.1 Root and top-level Concepts	191
6.1.2 Subtype Relationships	192
6.1.3 Defining characteristics	195
6.1.4 Qualifiers and refinement.....	198
6.1.5 Primitive and fully-defined Concepts	198
6.1.6 Important Concept Identifiers	198
6.2 Concept Model Specification.....	204
6.2.1 Hierarchies	204
6.2.2 Attributes Used in SNOMED CT	216
6.2.3 Miscellaneous Topics.....	250
6.3 Machine Readable Concept Model	251

7 Terminology Services Guide.....**253**

7.1 Representing SNOMED CT resources.....	253
7.1.1 Choosing a terminology server view.....	253
7.1.2 Choosing a technical approach.....	254
7.1.3 Example of a Full View Relational Representation.....	255
7.2 Importing SNOMED CT release data.....	260
7.2.1 Choosing a Release Type to import.....	260
7.2.2 Choosing the release files to import.....	275
7.2.3 Choosing extension files to import.....	277
7.2.4 Identifying release files using regular expressions	277
7.2.5 Checking during the import process.....	280
7.2.6 Pre-processing of distribution files by terminology server suppliers.....	281
7.3 Implementing Dynamic Snapshot Views	281
7.3.1 Generating Dynamic Snapshot Views	281
7.3.2 Optimizing Dynamic Snapshots Views.....	282
7.4 Working with metadata.....	284
7.4.1 Concept enumerations	285
7.4.2 Essential Reference Sets	291
7.4.3 Optional Reference Sets	297
7.4.4 Reference Sets supporting advanced functionality.....	298
7.4.5 Using other Reference Sets	299
7.5 Foundation Terminology services	300
7.5.1 Access to release information.....	300
7.5.2 Access to components.....	300
7.5.3 Access to essential concept Identifiers	305
7.6 User Interface Terminology services	306
7.6.1 Text Searches.....	306
7.6.2 Hierarchical Navigation.....	315
7.6.3 Applying Reference Sets	330
7.6.4 Access to qualifiers and refinable characteristics.....	332
7.7 Testing and traversing subtype Relationships	332
7.7.1 Top-level ancestor checking.....	333

7.7.2 Navigation concept checking.....	333
7.7.3 Subtype descendant testing.....	333
7.7.4 Subtype search scope restriction.....	333
7.7.5 Optimizing concept subsumption testing.....	333
7.8 Supporting Selective Data Retrieval.....	341
7.8.1 Creating queries.....	341
7.8.2 Types of queries.....	342
7.8.3 Creating legacy queries.....	401
7.9 Creating and maintaining Reference Sets	401
7.9.1 How to create a new Reference Set using an existing pattern.....	401
7.9.2 How to add, change or remove members of an existing Reference Set	408
7.9.3 How to create a new Reference Set pattern.....	410
7.10 Terminology Server Software.....	410
7.10.1 Terminology server functionality.....	410
7.10.2 Terminology server APIs	412

8 Record Services Guide.....**413**

8.1 Entering Expressions	414
8.1.1 Using text searches and subtype hierarchy navigation.....	414
8.1.2 Optimizing searches for data entry.....	414
8.1.3 Constraining searches for data entry.....	416
8.1.4 Constraining and extending hierarchical navigation for data entry.....	417
8.1.5 Constraining data entry.....	418
8.1.6 Configuring and applying data entry constraints.....	421
8.1.7 Entering qualifiers and other postcoordinated representations.....	422
8.1.8 Template and protocol driven data entry.....	424
8.2 Storing Expressions	425
8.2.1 precoordinated and postcoordinated representations.....	426
8.2.2 SNOMED CT storage issues for electronic health records	429
8.2.3 SNOMED CT storage options for effective retrieval.....	430
8.2.4 Record architectures, structures and semantics.....	432
8.2.5 Safely representing the context of recorded codes.....	434
8.3 Retrieval and Aggregation.....	437
8.3.1 Requirements for selective retrieval.....	437
8.3.2 Retrieving records containing selected concepts and their subtypes	439
8.3.3 Selective retrieval of postcoordinated expressions	441
8.3.4 Requirements for specific uses of selective retrieval.....	443
8.4 Communicating Expressions	444
8.4.1 Representation of SNOMED CT information in communications.....	444
8.4.2 Overlaps between SNOMED CT and Structural Semantics.....	445
8.4.3 Using Reference Sets to represent allowable value sets	446
8.4.4 SNOMED Clinical Terms and HL7.....	446

9 Change Management Guide.....	448
9.1 Managing Content Changes in SNOMED CT	448
9.1.1 Changes and historical notes.....	449
9.2 Managing Technical Changes in SNOMED CT	450
9.2.1 Release Format 2 Update Guide.....	450
9.2.2 RF1 Compatibility and Conversion Tools.....	470
9.2.3 SNOMED CT identifier Update Notes.....	471
9.3 Migrating Existing Data.....	473
9.3.1 Intended audience.....	473
9.3.2 Migration requirements.....	473
9.3.3 Strategies for migration.....	474
9.3.4 General considerations for data migration	474
9.3.5 Specific data migration issues.....	475
9.3.6 Migration from earlier SNOMED CT code systems.....	477
9.3.7 Migration from Read Codes and CTV3	478
10 Extension Services Guide.....	479
10.1 Rationale for Extensions	479
10.2 Extension Namespaces and SNOMED CT identifiers	479
Guidance for Producers of SNOMED CT Extensions.....	480
Guidance for Consumers of SNOMED CT Identifiers.....	483
Guidance where RF1 format is used for an Extension.....	485
10.3 Editing and Maintaining Extensions	485
11 SNOMED CT file and field names.....	487
A.....	487
B.....	488
C.....	488
D.....	489
E.....	490
I.....	490
K.....	491
L.....	491
M.....	491
O.....	492
Q.....	492
R.....	493
S.....	493
T.....	494
U.....	495
V.....	496
W.....	496

References.....498

12.1 SNOMED CT Background.....	498
12.1.1 SNOMED CT: A Comprehensive terminology for Health Care.....	498
12.1.2 Acknowledgments of Contributors to SNOMED CT®	500
HL7 version 3 - An object-oriented methodology for collaborative standards development.....	500
Desiderata for controlled medical vocabularies in the twenty-first century.....	500
HL7 Reference Information Model	501
Quality of clinical information retrieval using a semantic terminological model.....	501
Lexically Assign, Logically Refine strategy for integrating overlapping terminologies.....	501
Integration of tools for binding archetypes to SNOMED CT	501
Normal forms for description logic expressions of clinical concepts in SNOMED RT	502
Representing clinical information using SNOMED CT with different information models.....	502
Toward vocabulary domain specifications for health level 7-coded data elements.....	502

13 Glossary.....503

A.....	503
B.....	506
C.....	506
D.....	510
E.....	513
F.....	515
H.....	515
I.....	516
K.....	518
L.....	519
M.....	519
N.....	521
O.....	524
P.....	524
Q.....	526
R.....	527
S.....	530
T.....	535
U.....	537
V.....	538
W.....	538

Chapter

1

1 Preface



The purpose of this guide is to meet the needs of people who require an authoritative point of technical reference and advice to support their involvement in designing, developing, acquiring or deploying software applications that uses *SNOMED Clinical Terms*.

SNOMED Clinical Terms continues to evolve to further enhance its ability to represent clinical information and there is a growing body of knowledge about how best to use it. As a result future releases of the guide may include additional detail, revised advice and notes on significant changes to specifications.

This guide is available in two forms:

- **WebHelp (HTML):** (file suffix .html)
 - A hyper-linked version viewable in a standard web-browser.
 - This version includes searching and glossary lookups.
 - The web-based version is most effectively when used online. Some features may not work on a local version of this resource.
- **Adobe Acrobat:** (file suffix .pdf)
 - A browsable and printable version arranged for page layout rather than in separate topics.
 - The text content is identical to the HTML version but there are some difference to navigation and cross references resulting from page oriented formatting.
 - This version is searchable.

A version of each of the above is available configured for the US English and GB English. Note that the PDF versions are formatted for different paper sizes (US - Letter, GB - A4).

1.1 Notation used in this document



The following notation is used in this User Guide to represent key types of *SNOMED CT* information:

SNOMED CT concept names are generally represented using the *Fully Specified Name* in mixed case formatted as in the following example:

Example: | Peribronchial pneumonia (disorder) |

SNOMED CT attribute names are preceded and followed by a vertical bar. In some cases, to make them stand out they are presented in all capital letters as in the following example, though this is not to be considered a standard form for rendering *attribute names*:

Example: | FINDING SITE |

1.2 Document Properties



Table 1:

Title:	SNOMED CT Technical Implementation Guide
Date	2012-07-31
Version	2012-07-31 <i>International Release</i>
Creating Author:	David Markwell
Subject:	IHTSDO, Technical, Implementation Guide

1.3 Amendment History



Table 2: Amendment History

Version	Date	Editor	Comments
1.00	2002 02 31	David Markwell	Initial draft of TIG provided with first SNOMED CT release.
2.00	2002 07 31	David Markwell	SIEB 2002-06-13 agreed changes; support for moving <i>concept</i> between namespaces; additional status values; additional special <i>concepts</i> ; more consistent <i>Descriptions</i> of activity of <i>concepts</i> and <i>Descriptions</i> in relation to component status; addition of information on <i>Duplicate Terms Subset</i>
3.00	2003 07 31	David Markwell	Added inventory of documentation and updated copyright statements. Updated Cross References and Hyperlinks. Added Acknowledgments section. Added SNOMED CT Glossary. Added Distribution formats for the Practical Uses for Subsets sections from Subset Tables and Mechanisms technical specification document

Version	Date	Editor	Comments
4.00	2004 01 31	-	Release changes to reflect merger of Finding and Disease hierarchies
5.00	2005 01 31	-	Merged <i>Relationships</i> and <i>Historical Relationships</i> files “Part of” <i>Relationship</i> no longer a defining <i>Relationship</i>
6.00	2007 07 31	-	Updates to reflect transfer of IP to the <i>International Health Terminology Standards Development Organisation</i> . Removal of references to College of American Pathologists (CAP) derivative products
7.00	2008 07 31	-	Updated for the July 2008 <i>International Release</i> . <i>Release of References Table</i> . Updated diagrams and figures.
8.00	2010 01 31	-	Updated to reflect reclassification of Limited concepts as inactive.
9.00	2010 07 31	David Markwell	Major revision of content and format. Content migrated to <i>DITA</i> source files. Incorporated existing TIG and TRG material with addition of <i>Concept Model</i> Guides from User Guide. Initial outputs released in HTML and HTML Help format for review.
10.00	2011 01 31	David Markwell	Major updates based on feedback and quality review. <i>Release Format 2</i> specifications included in full. <i>Terminology Service Guide</i> updated to separate RF1 and RF2 advice. PDF format added to available publication forms.

Version	Date	Editor	Comments
11.00	2011 07 31	David Markwell	Various correction and minor revisions. Inclusion of update for <i>Concept Model</i> from User Guide source files. Additional advice on <i>Release Format 2</i> and <i>Transitive Closure</i> table generation. Gaps in glossary updated with draft glossary additions.
12.00	2012 01 31	David Markwell	Changes to accommodate revised policy on movement of components between <i>Extensions</i> and the <i>International Release</i> . Updates to OWL transform script notes (authored by Kent Spackman) Minor updates and corrections.

Version	Date	Editor	Comments
13.00	2012 07 31	David Markwell	<p>Removal of sections specific to RF1 (the original <i>SNOMED CT Release Format</i>) which was replaced in January 2012 by RF2. This changes has enabled some simplification of the structure of the document and a substantial reduction in size. The previous content of these sections is now published separately in the "<i>Release Format 1 Guide</i>".</p> <p>Creation of a separate DRAFT IHTSDO Glossary, which will eventually replaces the glossary items previous published as part of individual documents.</p> <p>Enhancement to the <i>Terminology services</i> Guide (in particular relating to "Supporting Selective Data Retrieval").</p> <p>Technical improvements for document structure and links to accommodate more effective referencing (particularly for the web published HTML Help version). The changes will support connections from forthcoming additions to the web-based documentation including an evolving set of Frequently Asked Question pages.</p>

1.4 Status



This guide contains parts and sections which differ in terms of the authority and status of their content. Each section of the guide is marked to indicate its publication type and status using the symbols shown in [Table 3](#) and [Table 4](#).

Table 3: Document Types

Type Name and Description	Draft	Review	Current
Standard A document or other resource that is intended to be authoritative. This includes specifications of <i>SNOMED CT</i> content and <i>release files</i> . Normative requirements for particular functions are also standards.			
Guidance A document or other resource that is intended to provide advice or suggest possible approaches to particular requirement or subject area.			

Table 4: Document Status

Status Name and Description	Standard	Guidance
Current Indicates that the document or resource is considered to be up-to-date and complete for the current release of <i>SNOMED CT</i> (indicated by an explicitly stated version date or by the publication date).		
Review Indicates that the document or resource has been released for review and comments from <i>SNOMED CT</i> users and other stakeholders. It is intended to be complete but has not been formally approved as a final version.		
Draft Indicates that the document or resource is a draft version. It may be incomplete and has not been approved in a final version.		

This edition of the document is configured to use US English .

The PDF version of this draft is formatted to be printed on US Letter paper.

Note: This is one of a several large documents that are regularly revised by the IHTSDO. Therefore, for the sake of the environment, please think carefully before deciding to print the entire document.

1.5 Referencing and Commenting



This document contains a way to reference topics a way that is not dependent on changes to the structure of the document as new versions are released including additional topics. These references are web addresses that will point to the latest version of and topic in the document.

If you are using the PDF version of the document there are three icons to the right of each title which provide useful information and relevant links.

-  The  icon indicates the status of the topic (see [Status](#)).
-  The  icon provides a link to the web address to accessing and referencing this topic online. Please use these reference to identify or share references to the topic as section and page numbers change between versions.
-  The  icon links direct to a page where you can submit comments or error report about this topic. The comment tracker is an online resource that requires you to login to an IHTSDO CollabNet. If you do not have an account, there is an option to create an account available on the login page.



If you are using the online web version of this document then there is a single bookmark icon  which, when clicked, opens a small form with an easy copy and paste option for access to the topic reference and button to click to take you direct to the comment tracker.

1.6 Additional information



Further information about *SNOMED CT* is available by contacting *IHTSDO*:

 **IHTSDO Contact Details:**

Web:

- www.ihtsdo.org

Email:

- support@ihtsdo.org

Address:

- IHTSDO
- Gammeltorv 4, 1.
- 1457 Copenhagen K
- Denmark
-
- Tel: +45 3644 8736
- Fax: +45 4444 8736

1.7 Inventory of Documentation



The following *SNOMED CT* documentation is made available to accompany the *International Release* of *SNOMED CT* from the International Health Terminology Standards Development Organization (*IHTSDO*). In the following listing hyperlinks are provided which will be maintained to point to the latest version of each of these documents.

- A list of documents, including a wider range of versions, is available from: www.ihtsdo.org/doc.

SNOMED CT Technical Implementation Guide (TIG)

- On line HTML version: www.ihtsdo.org/tig
- PDF version US English Letter page size: www.ihtsdo.org/tig.pdf
- PDF version UK English A4 page size: www.ihtsdo.org/tig_gb.pdf

The TIG is intended for *SNOMED CT* implementers, such as software designers. The TIG assumes information technology and software development experience. Clinical knowledge is not required, although some background is helpful to understand the application context and needs.

The TIG contains guidelines and advice about the design of applications using *SNOMED CT*, and covers topics such as *Terminology services*, entering and storing information, and migration of legacy information.

SNOMED CT Editorial Guide

- On line HTML version: www.ihtsdo.org/eg
- PDF version US English Letter page size: www.ihtsdo.org/eg.pdf
- PDF version UK English A4 page size: www.ihtsdo.org/eg_gb.pdf

The Editorial Guide is intended for clinical personnel, business directors, software product managers, and project leaders; information technology experience, though not necessary, can be helpful.

The Editorial Guide is intended to explain *SNOMED CT*'s capabilities and uses from a content perspective. It explains the content and *concept model*, and the principles used to edit the terminology.

SNOMED CT User Guide

- On line HTML version: www.ihtsdo.org/ug
- PDF version US English Letter page size: www.ihtsdo.org/ug.pdf
- PDF version UK English A4 page size: www.ihtsdo.org/ug_gb.pdf

The User Guide provides a less detailed introduction to the topics covered in the Technical Implementation and Editorial Guides.

IHTSDO Glossary (DRAFT)

- On line HTML version: www.ihtsdo.org/glossary
- PDF version US English Letter page size: www.ihtsdo.org/glossary.pdf
- PDF version UK English A4 page size: www.ihtsdo.org/glossary_gb.pdf

The Glossary is a general resource used to support all the other documents in this inventory.

SNOMED CT Release Format 1 Guide

- On line HTML version: www.ihtsdo.org/rf1
- PDF version US English Letter page size: www.ihtsdo.org/rf1.pdf
- PDF version UK English A4 page size: www.ihtsdo.org/rf1_gb.pdf

The RF1 Guide provides technical information relevant to those using the original *SNOMED CT Release Format*. Although this format was replaced by RF2 in January 2012, the old format is being maintained for a transitional period.

SNOMED CT Non-Human Refset Guide

- PDF version US English Letter page size: www.ihtsdo.org/guide/non_human_rs.pdf

A guide to use of the "Non-Human" Simple Reference Set that contains *concepts* and terms that are only used in veterinary medicine.

SNOMED CT Developer Toolkit Guide

- PDF version US English Letter page size: www.ihtsdo.org/guide/toolkit.pdf

A guide to use of value-added files and scripts that are provided as a toolkit available as part of the *SNOMED CT International Release*.

 **Additional Documentation:** The following materials previously published in separate documents are now integrated as part of the Technical Implementation Guide.

- *Technical Reference Guide*
- *Namespace Identifier Guide*
- *Namespace Identifier Registry*
- *File Naming Convention*
- *RF2 Data Structures Specification*
- *RF2 Reference Set Specifications*
- *RF2 Update Guide*
- *Stated Relationships Guide*
- *Canonical Table Guide* (previously included in RF1)

1.8 Where is the Glossary?



Some versions of documents may contain a glossary section. However, we are also developing a separate *IHTSDO Glossary* document which is currently available in a draft form. The intention is to move toward using this single common resource make it easier to ensure consistency across the *IHTSDO* community.

The current version of the *IHTSDO Glossary* is available as follows:

- On line HTML version: www.ihtsdo.org/glossary
- PDF version US English Letter page size: www.ihtsdo.org/glossary.pdf
- PDF version UK English A4 page size: www.ihtsdo.org/glossary_gb.pdf
- You can create links that query the glossary use the following web address pattern "www.ihtsdo.org/define/word or phrase". The following examples can be tested and you can include these types of reference in your documents to make it easy to refer to the *IHTSDO* glossary definitions:
 - www.ihtsdo.org/define/ihtsdo
 - www.ihtsdo.org/define/snomed ct
 - www.ihtsdo.org/define/affiliate licence

1.9 Copyright Notice



 **Copyright Notice:**

©2002-2013 The *International Health Terminology Standards Development Organisation (IHTSDO)*. All Rights Reserved. *SNOMED CT®* was originally created by The College of American Pathologists. "SNOMED" and "SNOMED CT" are registered trademarks of the *IHTSDO*.

SNOMED CT has been created by combining *SNOMED RT* and a computer based nomenclature and classification known as *Clinical Terms Version 3*, formerly known as *Read Codes Version 3*, which was created on behalf of the UK Department of Health.

This document forms part of the *International Release of SNOMED CT* distributed by the International Health Terminology Standards Development Organisation (*IHTSDO*), and is subject to the *IHTSDO's SNOMED CT Affiliate License*. Details of the *SNOMED CT Affiliate License* may be found at www.ihtsdo.org/licensing/.

No part of this document may be reproduced or transmitted in any form or by any means, or stored in any kind of retrieval system, except by an Affiliate of the *IHTSDO* in accordance with the *SNOMED CT Affiliate License*. Any modification of this document (including without limitation the removal or modification of this notice) is prohibited without the express written permission of the *IHTSDO*.

Any copy of this document that is not obtained directly from the *IHTSDO* (or a Member of the *IHTSDO*) is not controlled by the *IHTSDO*, and may have been modified and may be out of date. Any recipient of this document who has received it by other means is encouraged to obtain a copy directly from the *IHTSDO*, or a Member of the *IHTSDO*. (Details of the Members of the *IHTSDO* may be found at www.ihtsdo.org/members/).

Chapter

2

2 Overview of the guide



The overall structure of the guide is not intended as a suggested reading order but to provide a predictable location for each broad category of information. Thus as the guidance is extended and revised new sections will appear within the relevant locations rather than as separate documents.

The guide is divided into several parts each of which focuses on a particular aspect of technical implementation. The sections are interdependent and are extensively cross-linked. These links are used to avoid duplication and aid consistent maintenance.

- ***Structure and Content Guide (4)*** :
 - This part of the guide provides reference material on the technical design of *SNOMED CT*. These design features provide the foundation for the services described in subsequent parts of the guide.
 - The design features include: *SNOMED CT components* (*concepts*, *description* and *relationships*), extensibility mechanisms (*reference sets*), content *extensions* and *expressions* that are used to represent information within a electronic record.
- ***Release File Specifications (5)*** :
 - This part of the guide includes detail *Descriptions* of the files used to distribute *SNOMED CT* content to licensees;
 - The specification are an important source of technical reference for those developing and maintaining applications that provide access to *SNOMED CT* content.
- ***Concept Model Guide (6)*** :
 - This part of the guide describes the ways in which technical design of *SNOMED CT* is populated with content. The *Concept Model* specifies the main hierarchies in which *concepts* are arranged and the types of the *relationships* that are permitted between them.
 - This *Concept Model* is directly relevant to implementation because it determines the types of clinical ideas that can be expressed using *SNOMED CT* and the ways in which these ideas can be refined to represent more detailed information.
- ***Terminology Services Guide (RF2) (7)(RF1)*** :
 - This part of the guide describes the types of services required to access and make use of *SNOMED CT*. It also provides practical advice on effective ways to deliver these services based on practice experience.
 - These services include: importing distribution files, determining the *status* and properties of selected components, searching for *terms*, navigating hierarchies, testing and using *relationships* between *concepts*, working with references sets to determine *language* acceptability, membership of *value sets*, *cross maps* to other classification and additional *annotations* and metadata.
- ***Record Services Guide (8)*** :
 - This part of the guide describes the types of services required to use *SNOMED CT*, to represent instances of clinical information in *electronic health records*, knowledge resources, decision support algorithms and data retrieval specifications.

- These services include, entry of *expressions* (including *postcoordinated refinements*), storage of *expressions*, communication and selective retrieval of information that uses *SNOMED CT expressions* to represent clinical ideas.
- As part of the consideration of storage, communication and retrieval, this part of the guide also discusses the integration of the terminology with a well-designed information model. It is now widely recognized that this is crucial element in design and development of a *SNOMED CT enabled application*.
- *Record services* are dependent on *Terminology services* and these two sets of services may be tightly integrated. Alternatively, an application that delivers *record services* may use a third party *terminology server* to reduce the required development.
- ***Change Management Guide (9)*** :
 - This part of the guide addresses requirements that arise from changes to the content, structure and use of *SNOMED CT*.
 - The first significant change management challenge relates to *migration* from other coding schemes or from a less structured electronic record system. Decisions must be made about retaining or converting records, queries and protocols originally created using a terminology other than *SNOMED CT*.
 - Each release of *SNOMED CT* introduces some changes to content. From time to time there will also be changes designed to increase the expressivity of the *Concept Model*. Occasionally there may also be additional technical artifacts or specification developed to meet emerging requirements.
 - As systems evolve and as the content and structure of *SNOMED CT* are enhanced there is a continuing requirement to address to manage changes smoothly and without loss of information or functionality.
- ***Extension Services Guide (10)*** :
 - This part of the guide describes additional services which some advanced users or implementers may require to allow them to create or maintain *Extensions* for use in a particular country, organization or specialty.
 - The most common of these requirements will be to support the creation and maintenance of specialized *Reference sets*. Uses for *Reference Sets* include representation of *value sets*, marking *descriptions* to indicate acceptability of *terms* in a specific *language* or specialty, alternative hierarchies, *cross mapping* to classifications and *annotations*.

2.1 Who Should Read This Guide?



The guide can be used in various ways to assist the design, evaluation, operational implementation and use of various types of software applications that use *SNOMED CT*. The intended audience includes systems developers, health informatics specialists, purchasers, and system integrators.

2.1.1 Software designers and developers



- Software designers and developers should use this guide:
 - To enhance their technical understanding of *SNOMED CT* and the value it offers to their applications;
 - As a point of reference when designing a *SNOMED CT enabled application* and when planning and undertaking the required development.
- Designers and developers of fully integrated applications should use the guide:
 - As a checklist of *SNOMED CT* services necessary to meet the needs of their users;
 - For advice on how to implement the required services in ways that make the best use of *SNOMED CT* and which avoid known pitfalls.
- Designers and developers of *terminology servers* should use the guide:

- As a checklist when deciding which *SNOMED CT* services their server should offer;
- For advice on ways to implement the required services in ways that make the best use of *SNOMED CT* and avoid known pitfalls;
- As a point of reference when describing the functionality of their server.

- Designers and developers of applications that use *Terminology services* should use the guide:
 - As a checklist of *SNOMED CT* services necessary to meet the needs of their users;
 - To assist consideration of whether to use a *terminology server*;
 - As a point of reference when reviewing the functionality of *terminology servers*.

2.1.2 Health informatics specialists, analysts, purchasers and integrators



- Health informatics specialists, analysts, purchasers and integrators should use this guide:
 - To enhance their technical understanding of *SNOMED CT* and the value it offers to their organization ;
 - As a point of reference when specifying, procuring and evaluating *SNOMED CT enabled applications*.

- Health informatics specialists analyzing the needs of users and organizations should use this guide:
 - As a checklist of *SNOMED CT* services necessary to meet the needs of their users;
 - For advice on known pitfalls when implementing clinical terminologies;
 - To assist decisions on technical approaches to design and implementation of applications that use *SNOMED CT*.

- Purchasers of healthcare information systems should use this guide:
 - As a checklist when specifying procurement requirements for applications that use *SNOMED CT*;
 - As a starting point for the evaluation of the *SNOMED CT* related technical features of the available systems.

- Healthcare information systems integrators should use this guide:
 - As a checklist for confirming the claimed functionality of *SNOMED CT enabled applications*;
 - For advice on alternative approaches to integration of *SNOMED CT* related services into a wider information system.

- Information systems departments and project teams should use this guide:
 - As a checklist for the *SNOMED CT* related functionality needed to meet the requirements of their users;
 - For advice on alternative approaches to delivery and maintenance of *SNOMED CT* related functionality as part of an operational information system.

2.2 Important Notices



- 👉 Note:** The *IHTSDO* supplies *SNOMED CT* as a set of *release files* that are designed to be loaded into healthcare software applications such as *Electronic Health Records*. This guide describes services that should be provided by software applications that implement *SNOMED CT*.
- 👉 Note:** The *IHTSDO* does not create or market healthcare software applications but seeks to promote implementation and innovation by promoting a market place in which *SNOMED CT* is equally accessible to all software developers, vendors and health service providers.

👉 **Note:** This guide refers to files that are included in the *International Release of SNOMED CT* provided to licensees by the *IHTSDO*. It also refers to additional files that are included in *SNOMED CT Extensions* provided by *IHTSDO Members* and *Affiliates*. Details of the licensing arrangements for *SNOMED CT* and contact details for *IHTSDO Members* are available from the *IHTSDO* web site:

- www.ihtsdo.org

2.3 Additional information and feedback



Further information about *SNOMED CT* is available on the Internet at:

- www.ihtsdo.org

Please send feedback by email to:

- support@ihtsdo.org

or contact the International Health Terminology Standards Development Organisation at:

- IHTSDO
- Gammeltorv 4, 1.
- 1457 Copenhagen K
- Denmark
-
- Tel: +45 3644 8736
- Fax: +45 4444 8736

Chapter

3

3 SNOMED CT implementation



This part of the guide introduces the rationale for implementing *SNOMED CT*. It identifies some of the types of software application that benefit from the features of *SNOMED CT*. It sets out some broad parameters that determine the extent to which an application can make use of particular aspects of *SNOMED CT* and outlines some approaches to delivering the required services.

3.1 Motivation for Implementation



SNOMED Clinical Terms (SNOMED CT) is widely recognized as the leading global clinical terminology for use in *electronic health records*. It is maintained and developed by an International body (the *IHTSDO*) which has a growing community of Members and Affiliates. It is available free for use in *IHTSDO Member* countries and can also be used in other countries based on openly published licensing terms that are designed to be affordable. *IHTSDO* policies allow for the open involvement of its Members and *Affiliate Licensees* in the development of content and the design of future enhancements.

The features of *SNOMED CT* include:

- A broad scope that covers most of the clinical *concepts* used in patient centered clinical records;
- Ability to express different levels of clinical detail in patient record entries by using *expressions* containing one or more *concept identifiers*;
- *Relationships* between *concepts* that enable consistent retrieval of a common form of clinical information for many different purposes;
- Extensible design allowing graceful, evolutionary enhancement and addition of national, local or specialty content within a coherent standard structure;
- A *reference set* mechanism to support representation of *language / dialect* variants, value-sets, alternative hierarchies and mapping to classifications;
- *Component* permanence with history tracking;
- Good compliance with the essential features for future clinical terminologies as identified by *JJ Cimino* in his peer acclaimed 1998 paper.

SNOMED CT is designed to enable effective representation of clinical information in *electronic health records*. While there are other potential uses for *SNOMED CT*, the potential benefits are greatest where it is implemented as a part of a *Clinical Information System* centered on the delivery of health care services to individuals and populations.

The benefits actually realized by implementation depend on the technical design of applications and the way they integrate *SNOMED CT* with other essential elements. These technical issues are addressed in this guide. Another critical success factor is a process for managing implementation across an organization, region or country. Although the guide does not address broader issues of operational implementation within an organisation, it does provide a key source of reference for those specifying the practical details of a plan for large scale implementation of *SNOMED CT*.

3.1.1 Benefits for electronic health records



Implementation of *SNOMED CT*, as part of a well-designed *Clinical Information System*, is the key to unlock many of the potential benefits of *electronic health records*.

SNOMED CT enables consistent representation of clinical information within *electronic health records*. Its content and design allow most types of clinical information to be represented at levels of detail appropriate to a wide range of different use cases. The hierarchical and defining *relationships* of *SNOMED CT* facilitate effective meaning-based retrieval and reuse of this information. By using these *relationships*, a *SNOMED CT enabled application* can query *electronic health records* to extract, analyze and aggregate relevant data recorded in different settings and at different levels of detail.

Many of the benefits of *electronic health records* require an effective retrieval and reuse of clinical information. These include:

- Enhancing the care of individual patients:
 - Display of appropriate information to enable clinical staff to assess the condition and needs of patients;
 - Decision support tools that help to guide safe, appropriate and effective patient care;
 - Communicating, sharing and maintaining information in ways that enable different members of the health care team to access and use relevant information collected at different places and times.
- Enhancing the care of populations of patients:
 - Epidemiology monitoring and reporting;
 - Research into the causes of diseases;
 - Research into the effectiveness of different approaches to disease management and treatment.
- Supporting cost-effective delivery of care:
 - Using decision support to minimize the risk of costly errors in treatment;
 - Reducing duplication of investigation and interventions through effective access to shared information about the patient;
 - Auditing the delivery of clinical services; with more opportunity to analyze outliers and exceptions in the pattern of care delivery;
 - Planning future service delivery based on emerging health trends, perceived priorities and changes in clinical understanding.

Delivering these benefits depends on consistent representation of the various types of information that are represented in a health record. It must be possible to represent this information at different levels of detail and it must be possible to query this information from various perspectives and at different levels of detail. To meet these requirements *electronic health records* need a well-maintained terminology that meets the criteria specified in *Desiderata for Controlled Medical Vocabularies in the Twenty-First Century* (Cimino JJ in *Methods Inf Med.* 1998 Nov;37(4-5):394-403). *SNOMED CT* addresses these requirements and additional practical requirements for an implementable, globally applicable but locally extensible, multilingual solution.

3.1.2 Benefits for knowledge representation



Implementation of *SNOMED CT* within a knowledge resource, such as an electronic reference book, clinical guideline, decision support protocol, facilitates effective access from, or integration with, *Clinical Information Systems*.

The use of *SNOMED CT* in *electronic health records* enables consistent processable representation of clinical information. Potential uses of this information include linkage to knowledge sources to assist its understanding and interpretation.

Developers of decision support protocols, care pathways or data analysis packages can benefit by using *SNOMED CT* to represent requirements for clinical information collection and processing. This allows direct

translation of the protocol into queries that can be applied directly to a *SNOMED CT* enabled *electronic health record*.

Publishers of knowledge based resources can benefit by tagging their information using *SNOMED CT*. These tags can be used to index information by *concept* rather than by *keywords*. As a result, relevant information can be identified by users during interaction with an *electronic health record*. For example, when selecting a particular item during data entry or review potentially relevant articles can be listed and/or displayed.

SNOMED CT also offers benefits during the development of knowledge resources. Tagging information using *SNOMED CT* while authoring knowledge artifacts may identify potential ambiguities that would otherwise be overlooked.

3.1.3 Benefits of an open global approach



Implementation of *SNOMED CT* offers the benefit of a global approach to the requirements for clinical terminology.

Any country or large organization that is developing or deploying *electronic health records* needs to consider the requirements for consistent representation of clinical information. One element of the solution is usually a coding scheme, controlled vocabulary or terminology. The breadth or scope and depth of detail in clinical records means that the set of codes or *terms* required is large and grows rapidly as additional disciplines and specialties become involved. Similarly the interdependency of *terms* used in different domains leads to a significant level of complexity.

Developing and maintaining a terminology that adequately addresses clinical requirements is a substantial task. A global approach has significant benefits by enabling economies of scale for National bodies and health care service providers.

A global approach also encourages common solutions to some of the challenges posed by requirements for consistent representation of complex information. The resulting reduction in divergence provides a more secure foundation for implementers who wish to deploy their applications in many countries.

Implementing a global clinical terminology also enables applications to be deployed in other countries without needing to switch between terminologies. It also allows use of other standards and materials that incorporate or are designed for use with that terminology. The ability to integrate components and standards based on a common terminology is a major advance over solutions that depend on a local or proprietary code system.

A global clinical terminology also provides a foundation for communication and sharing of information. The information communicated may include clinical records used to support delivery of health care to a mobile population. It may also include aggregations of records used for epidemiology and multi -center research.

3.1.4 Benefits of extensibility and configurability



Implementation of *SNOMED CT* allows common approaches to be applied to extend and configure the terminology for use in a particular environment.

Most clinical *concepts* are relevant in all countries, organizations and specialties but some *concepts* are relevant only to a particular environment. *SNOMED CT* allows national, local or organizational requirements to be addressed by separately maintained *SNOMED CT Extensions*. *SNOMED CT enabled implementations* can benefit from the content in these *Extensions* without the need for any additional software development because *Extensions* have exactly the same structure as the *International Release*.

SNOMED CT covers a broad domain to depth of detail appropriate to a range of health care disciplines and clinical specialties. As a result, it has an extensive content, different parts of which are needed in particular environments. The *SNOMED CT* design includes the *Reference Set* mechanism which provides a standard way to refer to a set of *SNOMED CT components*. *Reference Sets* can be used to configure different views of *SNOMED CT* by constraining searches or representing short lists of terms for a data entry field. They can also be used to meet other requirements including checking that a *concept id* falls within a permitted set of values for a field in a data structure or message (e.g. to represent an *HL7 value set*).

- Organizations implementing *SNOMED CT* benefit from *Reference Sets* because they allow requirements for use of particular terms and *concepts* to be represented in a form that can be applied to any *SNOMED CT enabled application*. This allows *Reference Sets* to be shared throughout and between organizations , even when different software is used to meet local or departmental requirements.
- Software developers and vendors benefit because *Reference Sets* provide a common, machine processable representation of requirements for different patterns of use of *SNOMED CT*. This simplifies local configuration and enhances interoperability with other *SNOMED CT enabled applications*.

3.2 Implementation Types



SNOMED CT itself is only a part of the solution to addressing the requirements for effective electronic clinical records. A terminology on its own "does" nothing unless it is implemented as part of an application and used. Implementation of *SNOMED CT* requires software applications that exploit its features to meet the real and perceived needs of users.

The "users" of *SNOMED CT* include:

- Those who specify, commission and configure software for use in a particular clinical environment;
- End-users who enter or retrieve clinical information.

As illustrated by *Figure 1*, users experience *SNOMED CT* through application software which delivers services to access and apply *SNOMED CT*. The ways in which applications apply the features of *SNOMED CT* to address user requirements determine the extent to which the potential benefits are realized .

The following sections summarize some of the types of implementation that may be needed to meet different requirements. Some types of application do not need to support or use all *SNOMED CT* features. However, there are some overarching requirements for consistency between implementation used within a given organization , country or region. Even where requirements are limited, care should be taken to ensure that *SNOMED CT enabled applications* are aligned with good practice and with agreed policies applicable to the situations in which they are used.

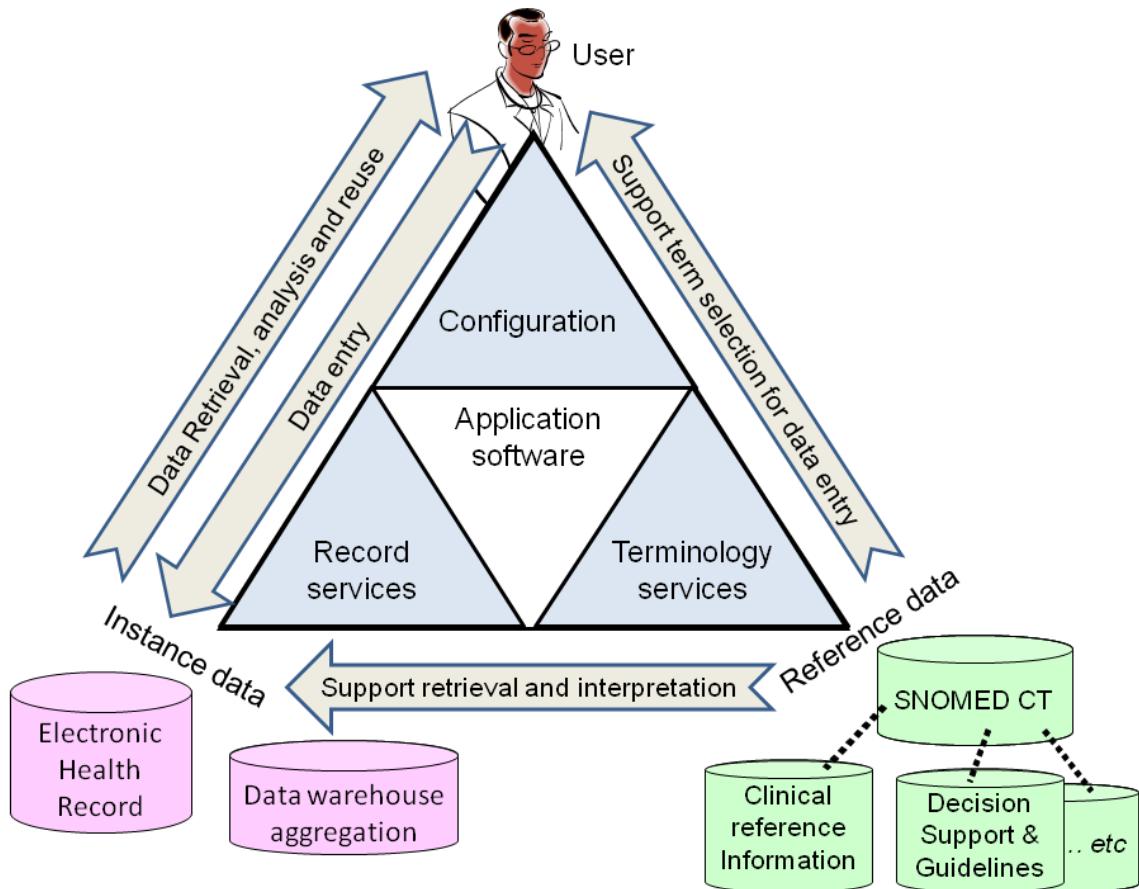


Figure 1: Relationship between users application software and SNOMED CT

3.2.1 Implementation Types - Clinical records



A *SNOMED CT* enabled clinical record application uses *SNOMED CT expressions* to represent clinical information in the records of individual patients.

Clinical record applications include specialized departmental systems, organization-wide systems and systems that integrate multiple systems to deliver a distributed *electronic health record* or a collection of widely accessible summary records.

A *SNOMED CT* enabled clinical record application needs to provide *record services* including entry, storage, retrieval and communication of *SNOMED CT expressions*. These *record services* depend on *terminology services* including the ability to search for *concepts* and to interpret stored *SNOMED CT expressions*.

A wide range of types of information can be represented at different levels of detail using *SNOMED CT expressions*. The types of information and level of detail that are used may vary depending on user requirement or may be limited by the design of the application. Differences in the required level of expressivity influence the range of *record services* that need to be supported.

The way that *SNOMED CT expressions* are represented within a record structure affects the range of services that are required to deliver the potential benefits of implementation. The value of the rich expressivity of *SNOMED CT* may be enhanced or diminished by the way the record structure relates *SNOMED CT expressions* to surrounding contextual information. For example, if a record structure permits similar or related information to be recorded in several ways a query to retrieve that information will need to consider all these possibilities. Retrieval is simpler if similar information is recorded in a consistent way - irrespective of the way it was entered. This issues are discussed in detail in the [Record Services Guide \(8\)](#).

3.2.2 Implementation Types - Knowledge representation



A *SNOMED CT* enabled knowledge representation uses *SNOMED CT expressions* to represent or tag resources that represent clinical knowledge. Examples of resources that can be *SNOMED CT* enabled include electronic reference books, clinical guidelines, care pathways, decision support protocols and requirements for analysis and audit.

There are various ways in which *SNOMED CT expressions* can be used in a knowledge resource. These can be divided into two broad categories:

- Use of *SNOMED CT expressions* as an integral part of a structured representation of knowledge:
 - For example, a decision support rule that tests for the existence of a record of a particular type of finding represented using a *SNOMED CT expression*.
- Use of *SNOMED CT expressions* to tag or index a knowledge resource:
 - For example, a reference book in which textual *descriptions* of indications, contraindications and side effects of particular treatments are tagged with *SNOMED CT expression* that can be used to allow context-sensitive retrieval of relevant information.

There are two distinct but interrelated aspects to *SNOMED CT* knowledge representation.

- Applying *SNOMED CT expressions* to the resource:
 - The form of representation to be used must be specified in a way that takes account of the ways in which the resource is to be used and accessed.
 - The knowledge authoring environment must allow the specified representation to be applied consistently. This requires use of *Terminology services* that allow searching and selection of *concepts*. Depending on the level of detail required, there may also be a requirement to support the construction of *postcoordinated expressions*.
- Enabling appropriate access to and use of the resource:
 - The types of access required depend on the intended functionality.
 - The most basic level of functionality involves using *SNOMED CT expressions* as a *concept*-based index. By taking account of the *SNOMED CT subtype hierarchy* and defining *relationships*, a *concept*-based index can provide more relevant results than a simple *term* based search.
 - More sophisticated uses such as clinical decision support require *SNOMED CT expressions* in the knowledge resource to be used to generate queries that can be applied to information stored in an *electronic health record*.
 - The provider of a *SNOMED CT* enabled knowledge resource may provide a specification that allows software developed by other organizations to interrogate it and provide the required level of functionality. Alternatively, the knowledge authoring organization may also develop and provide software that delivers the intended functionality.

3.2.3 Implementation Types - Aggregation and analysis



SNOMED CT enabled aggregation and analysis systems use *SNOMED CT* to enable effective aggregation and analysis of information derived from clinical record systems.

SNOMED CT enables consistent processable representation of clinical information. As well as presenting opportunities for analysis of information within an individual clinical record system, this can be used to support analysis of a broader substrate of aggregated data.

There are two types of approach that can be employed to enable analysis of aggregate data.

- A *SNOMED CT* enabled data warehouse:

- The content and structure of data required from individual *Clinical Information Systems* is specified. The specified structure must include details of the required representation of data including *SNOMED CT expressions*.
- The required data is extracted and uploaded to a database designed for the purpose of large scale analysis. Usually the extract and upload will need to be repeated or updated at specified intervals.
- The central database is structured to optimize common types of queries taking account of the *SNOMED CT expressions* and the *relationships* between referenced *concepts* asserted in *SNOMED CT* content.
- A *query interface* is provided to allow common types of question to be expressed against the central database.
- Queries are run taking account of the *relationships* between *concepts* to provide comprehensive and accurate results (minimizing the risks of false negatives or false positives).
- The results of queries are presented where relevant using *SNOMED CT expressions* as processable labels to enable further analysis.
- A common *query specification* supported by clinical record systems:
 - A common *reference information model* including *SNOMED CT expressions* is specified. This is used as a common *model of meaning* against which queries are evaluated.
 - Each clinical record system provider implements this common *model of meaning* as a *view* of the information stored in their *electronic health record*.
 - A *query interface* is provided to allow common types of question to be expressed against the common *model of meaning*.
 - Queries are distributed and run on individual systems and the results are returned to a central system for aggregate reporting.
 - The results of queries are presented where relevant using *SNOMED CT expressions* as processable labels to enable further analysis.

In practice, there is significant overlap between these two approaches. A data warehouse approach can benefit from a common approach to specifying the information extraction requirements. This allows changes to the specification without re-engineering the contributing clinical record systems. A common *query specification* approach requires a central element to manage distribution of queries and aggregation of results.

Irrespective of the approach taken, *SNOMED CT* enabled aggregation and analysis is most effective where the representation of information in the contributing clinical record systems is consistent with a common view. However, it is possible to aggregate information from diverse systems if the limits imposed by differences are understood. It is even possible for a *SNOMED CT* aggregation and analysis system to be applied to information that was not originally encoded using *SNOMED CT*. An extraction and aggregation interface that includes mapping from another coding system may produce information of adequate quality and consistency for many purposes. Data derived by tagging textual records using *natural language processing* may also meet requirements that are not safety-critical.

3.2.4 Terminology tools



SNOMED CT enabled terminology tools provide access to *SNOMED CT* content. On their own they are not practical end-user implementations but they enable the development and review of *SNOMED CT*. They may also deliver services that can be used by end-user implementations.

3.2.4.1 Implementation Types - Terminology browser



A *SNOMED CT* enabled *browser* allows the content and structure of *SNOMED CT* to be explored and reviewed.

A typical *SNOMED CT* enabled *browser* can locate *concepts* and *descriptions* by *Identifiers* and by searching the text of *description terms*. Various views of located *concepts* may be displayed including the set of related *descriptions*, the hierarchical *relationships* and other defining *relationships*.

A terminology *browser* may be:

- A stand-alone tool.
- Part of a more extensive implementation.
- Accessible via an *Application Programming Interface (API)*:
 - This may allow the *browser* to be used by client applications to select *SNOMED CT expressions*;
 - It may be part of a *terminology server* which provides a wider range of *Terminology services*.

3.2.4.2 Implementation Types - Terminology server



A *SNOMED CT enabled terminology server* is a software application that provides programmatic access to *SNOMED CT components*. These services are made available through a documented *Application Programming Interface (API)* which can be used by many different client applications.

A *SNOMED CT enabled terminology server* must be able to *import SNOMED CT release files* and provide some or all of the services described in the *Terminology Services Guide (7)*. All *terminology servers* must support a basic minimum set of functions including *Foundation Terminology Services* and access to *Reference sets and other metadata*.

A *terminology server* may provide *user interface* services, such as a set of screen controls to support term selection. Alternatively, while the *API* should support searches, the *user interface* representation of the results of a search may be left to client applications. Where *user interface* controls are provided by the server, these controls may also be packaged in an integrated form as a *terminology browser*.

A *SNOMED CT enabled terminology server* may also provide services that support the use of other terminologies. In this case, it may conform to a standard specification such as *Common Terminology Services 2 (CTS2)*.

3.2.4.3 Implementation Types - Terminology development and maintenance tools



SNOMED CT development and maintenance requires tools which are able to create and update *SNOMED CT* content.

Development and maintenance tools may either be general purpose or may focus on specific requirements (e.g. *Reference Sets* to support *language*, *cross mapping* or development of value-sets).

The process of maintenance needs to track changes and manage conflicts between edits made by different authors. In the case of content development, the tools must also ensure that *concept definitions* conform to the *SNOMED CT Concept Model*. At regular intervals the tools need to generate a consistent set of quality assured *release files*.

The *IHTSDO Workbench* is a set of software tools designed to support the development, maintenance, and use of *SNOMED CT*. Its key role is to facilitate the maintenance of the *SNOMED CT International Release* and the *National Extensions* developed by *IHTSDO Members*. However, the future scope of use may extend to other organizations and to health information systems around the world. The *Workbench* is owned by the *IHTSDO* and is available under an Open Source license agreement.

3.3 Implementation Levels



SNOMED CT can be implemented in a wide range of clinical record applications. These include systems developed for use with other code systems that have been adapted to support *SNOMED CT* as well as systems designed with the assumption that *SNOMED CT* would serve as the primary terminology. The *SNOMED CT* features that applications support and use may vary, partly due to differences in user requirements and partly due to development priorities. Against this background of variability, it is reasonable to ask what is a *SNOMED CT implementation* or what is a good *SNOMED CT implementation*. While there is not a single or simple answer to these questions, this section identifies some key dimensions which determine the capability of *SNOMED CT* enabled clinical record systems.

Each of the following sections describes a dimension and outlines a spectrum of capabilities ranging from absence of support (Level 0) to full support (Level 2). A mixture of Level 0 and Level 1 capabilities are likely to be found in existing systems that have been adapted to work with *SNOMED CT*. A system specifically developed to work with *SNOMED CT* should be expected to have capabilities that are at least at the high end of the Level 1 spectrum and should ideally have Level 2 capabilities.

The specification of different levels is not intended to suggest a step-by-step development path. Those needing to rapidly *SNOMED CT* enable an existing clinical record system are recommended to follow a two stage approach.

1. Design, develop and deploy a revision to the current system to support Level 1 capabilities that meet known short or medium term requirements:
 - The level achieved in this stage will depend on customer requirements and the design limitation of the existing system.
2. Design and develop a new or substantially revised system (including revised record structures) to support a mixture of high-end Level 1 and Level 2 capabilities:
 - The level at which this development is target should be one that meets anticipated medium to long term requirements;
 - Even if the initial target of the work is limited to the high-end of Level 1, the design should be sufficiently flexible to enable Level 2 capabilities to be added when required.

Developers who do not require a rapid deployment based on a revision of an existing systems are recommended to skip the first step and proceed to design and develop a flexible solution that utilizes the key strengths of *SNOMED CT*.

Each of the following sections describes one dimension that contributes to the overall implementation level. It is important to recognize that:

- This is not a formal scoring scheme:
 - Some dimensions are more significant than others;
 - The significance of reaching a particular level depends on the nature of the application and the user requirements it seeks to address.
- Many of the dimensions are inherently interdependent:
 - For example, Level 2 data entry capabilities are not compatible with Level 1 data storage.

3.3.1 Implementation Level - Scope of use



A clinical record system may use *SNOMED CT expressions* to represent some or all of the types of information outlined in the list below. The types of information for which *SNOMED CT* can be used may be limited by the structure used to store the *electronic health record*. The significance of these limitations depends upon the intended use of the clinical record system.

- Level 0: No support for *SNOMED CT expressions*.
- Level 1: Support for use of *SNOMED CT* limited to particular types of clinical data:
 - Addressing the requirements for a particular type of use;
 - Addressing a set of requirements specified by a particular organization .
- Level 2: Support for consistent use of *SNOMED CT* across a broad scope of information types:
 - Providing a general purpose approach to the use of *SNOMED CT* within an *electronic health record*
 - Allowing configuration to vary the scope of coverage to meet specific requirements.

The following check-list identifies some of the *electronic health record* elements in which *SNOMED CT expressions* might be used. The list is not complete but it covers many of the areas in which use of *SNOMED CT* has been discussed in *IHTSDO* working groups. It is intended to assist consideration of the areas in which *SNOMED CT* should be used to meet the needs of users and organizations . The inclusion of an item in this list does not imply that the *SNOMED CT International Release* provides comprehensive content to populate that part of the record.

1. Disorders, diagnoses and problems:

- Problem list entries;
- Admission diagnosis;
- Discharge diagnosis;
- Provisional or working diagnosis;
- Differential diagnosis.

2. Symptoms:

- Presenting symptoms;
- History of current condition;
- Other symptoms.

3. Allergies and adverse reactions:

- Adverse reaction events;
- Allergies and other propensities to adverse reactions.

4. Procedures:

- Operative procedures.
- Diagnostic procedures.
- Medications:
 - Current medication;
 - Prescriptions;
 - Dispensing records;
 - Drug charts.
- Other therapeutic procedures:
 - Other therapy requests;
 - Other therapy delivery and outcomes.

5. History:

- Medical and surgical past history;
- Medication history;
- Family history.

6. Examination findings:

- Vital signs;
- Clinical examination findings.

7. Investigation information:

- Laboratory investigations:
 - Laboratory investigation requests;
 - Laboratory investigation procedures;
 - Laboratory investigation results.

- Diagnostic imaging:
 - Diagnostic imaging requests;
 - Diagnostic imaging procedures;
 - Diagnostic imaging results.
- Other investigations:
 - Other investigation requests;
 - Other investigation procedures;
 - Other investigation result.

8. Other types of clinical information:

- Planned actions;
- Risk, goal and expected outcomes;
- Scale based assessments;
- Progress notes.

9. Administrative information:

- Admission, transfer and discharge events.

10. Other values:

- Body sites, structures and locations;
- Organisms;
- Substances (other than drugs);
- Pharmaceutical and biological products (drugs).

3.3.2 Implementation Level - Record structure



The logical model underlying the structure of the record has a direct effect on the ability of a *SNOMED CT* enabled clinical record system to take advantage of the features of *SNOMED CT*. An application may use an optimized proprietary internal representation of the *electronic health record*. However, consistent use of *SNOMED CT* across a range of applications requires a common reference model to which proprietary structures are mapped. In addition to this, the ways in which *SNOMED CT expressions* are used within a common *reference information model* need to be constrained to improve predictability and minimize ambiguity.

- Level 0: A proprietary structure that is neither aligned with nor mapped to a standard *reference information model*.
 - Low: Text only record with no use of clinical codes;
 - High: Structured record supporting use of clinical codes.
- Level 1: A structure that is aligned with or mapped to a standard *reference information model*.
 - Low: Proprietary structure mapped to a standard model to support limited messaging requirements. Supports the use of *SNOMED CT* coding within that structure.
 - High: Structure aligned with a standard *reference information model* that supports that supports use of *SNOMED CT* coding.
 - Examples of standard *reference information models* include:
 - The *HL7 Version 3 Reference Information Model* (RIM);
 - The *CEN TC251 Health informatics - Electronic health record communication - Part 1: Reference model (EN13606)*.

- Level 2: An aligned or mapped structure in which *SNOMED CT expressions* are used in accordance with agreed guidelines for use of a standard *reference information model*:
 - In Level 2 *SNOMED CT* is used in accordance with *terminology binding* guidance to minimize the semantic gaps and overlaps between the terminology and the information model. Without constraints, these gaps and overlaps lead to inconsistent representation of similar data and thus limit the effective reuse of information.
 - Example of agreed guidelines for using use of *SNOMED CT expression* in particular reference models include:
 - The *HL7TermInfo DSTU - Guide to the Use of SNOMED CT in HL7 Version 3*;
 - Guidance on *terminology binding* developed by the *UK NHS Logical Record Architecture* for use in an *EN13606* based logical model.

3.3.3 Implementation Level - Expression storage



Support for storing *precoordinated* and *postcoordinated SNOMED CT expressions* determines the extent to which *SNOMED CT* can be used to represent detailed information within an *electronic health record*.

- Level 0: No support for storage of *SNOMED CT expressions*
- Level 1: Support for storage of *precoordinated SNOMED CT expressions*:
 - Support for storage of a *precoordinated expression* implies the ability to store a representation of a *concept identifier* as part of each item for which *SNOMED CT* is used:
 - The *concept identifier* may be represented as a 64-bit *integer* or as an 18-digit *string*;
 - Other internal representations may be used provided they can be resolved to the appropriate *Identifier* for display, communication or processing.
- Level 2: Support for storage of *postcoordinated SNOMED CT expressions*:
 - Support for storage of *postcoordinated expression* implies the ability to store a representation that captures the logical model of a *postcoordinated expression*:
 - The simplest reference representation is the *SNOMED CT compositional grammar* which provides a *string* representation. Due to the open-ended nature of the *postcoordinated strings* are of indeterminate length.
 - The guide discusses alternative representations including the use of *expression reference table* which enables use of a fixed length reference within the records. This approach uses a *UUID* which can be represented as a 128-bit *integer* or as a hexadecimal *string*.
 - This level has variants depending on the extent of support for *postcoordinated expression* storage:
 - Low: Storage of *postcoordinated expressions* limited to specific fields in the record structure;
 - High: Full support for storage of *postcoordinated expression* allowing any valid *expression* to be stored and retrieved.

3.3.4 Implementation Level - Data entry



The categorization in this section is based on the extent to which the system enables entry of *SNOMED CT expressions*. In addition, this section indicates the importance of a well-designed user-interface.

- Level 0: No support for entry of *SNOMED CT expressions*.
- Level 1: Support for *precoordinated SNOMED CT expression* entry:

- Low: Access limited to fixed set of *SNOMED CT concepts*;
- Medium: Access to full content of *SNOMED CT*;
- High: Access to full content of *SNOMED CT* with configurable value-sets matched to user requirements.
- Level 2: Support for *postcoordinated expression* entry:
 - Low: Access to limited *postcoordination* (matching data storage restrictions);
 - Medium: Access to full range of *postcoordination* supported by the *Concept Model*;
 - High: Access to *postcoordination* with configurable *constraint* matched to user requirements.

Another important data entry issue is the ease of use which depends on the usability, relevance and performance of searches. Where *postcoordinated* data entry is supported the approach to selecting or constructing *postcoordinated expressions* is also significant.

An attempt to categorize specific approaches to the user-interface is subjective as alternative *user interfaces* may be appropriate to different uses. However, for most environments a flexible range of configurable *SNOMED CT* aware user-interface tools is likely to offer a better user experience than reliance on a one-size fits all *browser* or search engine.

3.3.5 Implementation Level - Data retrieval



A major strength of *SNOMED CT* is its ability to support meaning based selective retrieval. The extent to which this feature is used by a clinical record system determines the value of entering and storing the data.

- Level 0: No native support for *SNOMED CT* enabled data retrieval:
 - This level has variants depending on whether it can map code in exported data to *SNOMED CT expressions*:
 - Low: No support for *SNOMED CT* based analysis;
 - High: Support for extracting a specified set of locally coded data and mapping the local codes to appropriate *SNOMED CT expression* for central aggregation and analysis.
- Level 1: Support for retrieval of *precoordinated SNOMED CT expressions* :
 - This level has a spectrum of variants depending on the level of support for the following features:
 - *Query expressivity*: The ability to express *query* predicates that explicitly include or exclude *subtypes* of specifically identified *concepts*;
 - *Subsumption testing*: Use of *SNOMED CT subtype hierarchy* to interpret and evaluate queries;
 - *Concept Equivalence*: The ability to retrieve equivalent information even if it is represented in different structures within the record;
 - *Context awareness*: The ability to take account of contextual information, derived from the record structure and/or the *SNOMED expression*, when interpreting and evaluating queries;
 - *Performance*: The ability to interpret and evaluate queries within an appropriate period of time and without causing deterioration in other system functions.
- Level 2: Support for retrieval of *postcoordinated SNOMED CT expressions* :
 - This level has a spectrum of variants depending on the level of support for the following additional aspects of the features specified for Level 1:
 - *Query expressivity*: The ability to represent *postcoordinated* predicates in a *query*;
 - *Subsumption testing*: Use of *defining characteristics* and *normal form transformations* (or a *description logic classifier*) to determine whether *expressions* are subsumed by *query* predicates;
 - *Equivalence*: Use of *defining characteristics* and *normal form transformations* (or a *description logic classifier*) to determine *equivalence* between different *postcoordinated expressions* and in different structures within the record;

- Context awareness: The ability to take account of contextual information derived from the record structure and/or *postcoordinated SNOMED expressions*, when interpreting and evaluating queries;
- Performance: The ability to interpret and evaluate queries that support *postcoordinated representations* within an appropriate period of time and without causing deterioration in other system functions.

3.3.6 Implementation Level - Communication



The ability to send and receive *SNOMED CT expressions* in messages or other communication is partially dependent on data entry, storage and retrieval capabilities. However, some types of communication may be supported by mapping or human-readable renderings even in the absence of internal support for *SNOMED CT*.

- Level 0: Mapping based support for communication of *SNOMED CT expressions* :
 - Inbound communications containing *SNOMED CT expressions* :
 - Low: Not supported.
 - Medium: Rendered as human-readable text. Unless the inbound message also contains the *term* text, this requires access to some *SNOMED CT* enable *Terminology services* to lookup and display the relevant *terms*.
 - High: Mapped to an internal coding scheme or classification. This may be feasible to support specific use cases but not for the full scope of clinical information.
 - Outbound communication containing *SNOMED CT expressions* :
 - Low: Not supported;
 - Medium: Supported for a few specific types of clinical data in the existing system by mapping to from an existing code system to *SNOMED CT*;
 - High: Supported for most clinical data in the existing system by mapping to from an existing code system to *SNOMED CT*.
- Level 1: Native support for communication of *precoordinated SNOMED CT expressions* :
 - Inbound communications containing *precoordinated SNOMED CT expressions* :
 - Low: Supported for some types of information but constrained by data entry and *expression storage* capabilities;
 - High: Supported for most types of information.
 - Outbound communications containing *precoordinated SNOMED CT expressions* :
 - Low: Supported but limited by data entry and storage and retrieval capabilities;
 - High: Supported for most types of information.
- Level 2: Native support for communication of *postcoordinated SNOMED CT expressions* :
 - Inbound communications containing *postcoordinated SNOMED CT expressions* :
 - Low: Support limited to particular attributes (e.g. |l laterality|, |causative agent|) in *postcoordinated expression*;
 - Medium: Support for general *postcoordination* applied to some types of information;
 - High: Able to receive, process and store any valid *postcoordinated expression*.
 - Outbound communications containing *postcoordinated SNOMED CT expressions* :
 - Low: Support limited to particular attributes (e.g. |l laterality|, |causative agent|) in *postcoordinated expression*;

- Medium: Support for outbound communication of any *postcoordinated expression* that can be entered or stored in the system;
- High: Support for outbound communication of any valid *postcoordinated expression*.

3.4 Implementation Services



When designing or implementing a *SNOMED CT enabled application*, the first step is to assess the range of services necessary to meet user requirements. The two main categories of services required by applications are *terminology services* that only interact with the terminology and *record services* which apply the terminology to instance data. These services are described in separate sections of this guide.

The [Terminology Services Guide](#) describes services that access *SNOMED CT* reference data. These services are summarized in [Figure 2](#).

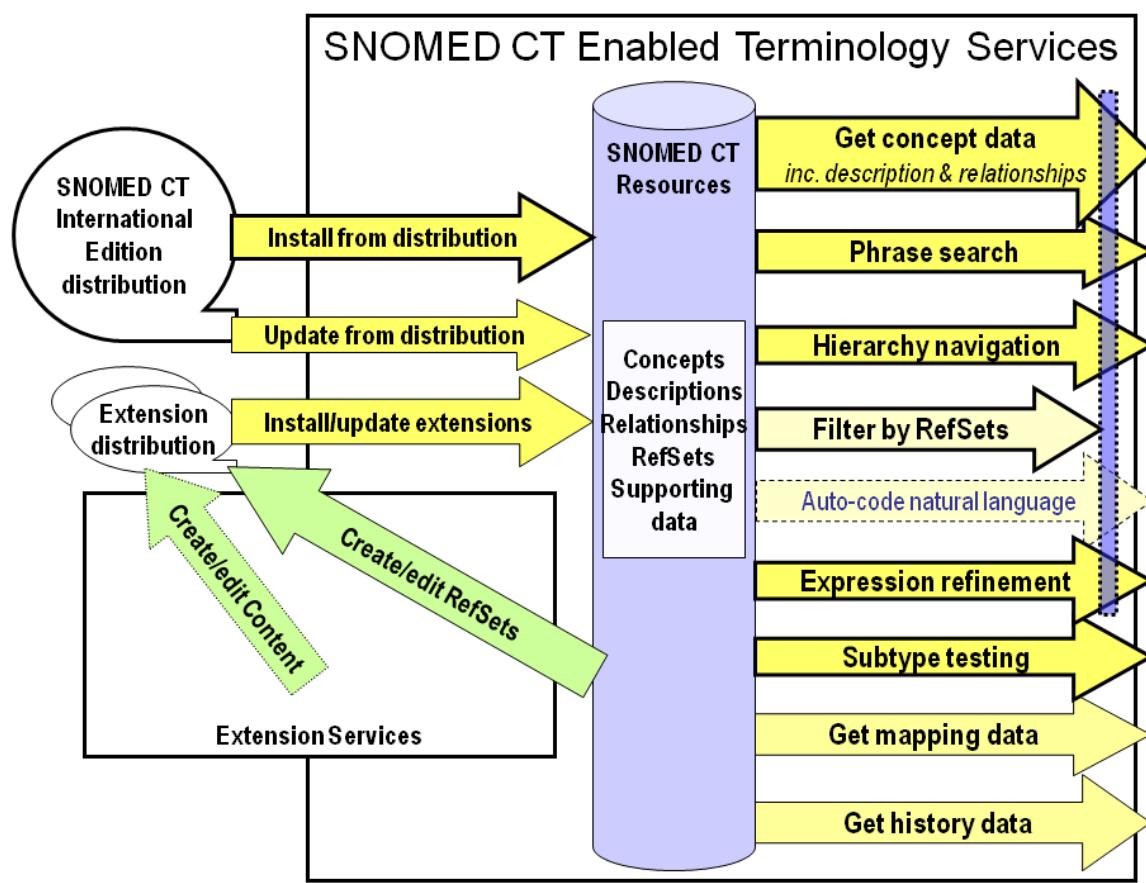


Figure 2: SNOMED CT Enabled Terminology services

The [Record services guide](#) (8) describes services that apply *SNOMED CT* to represent information in a clinical record. These services are summarized in [Figure 3](#).

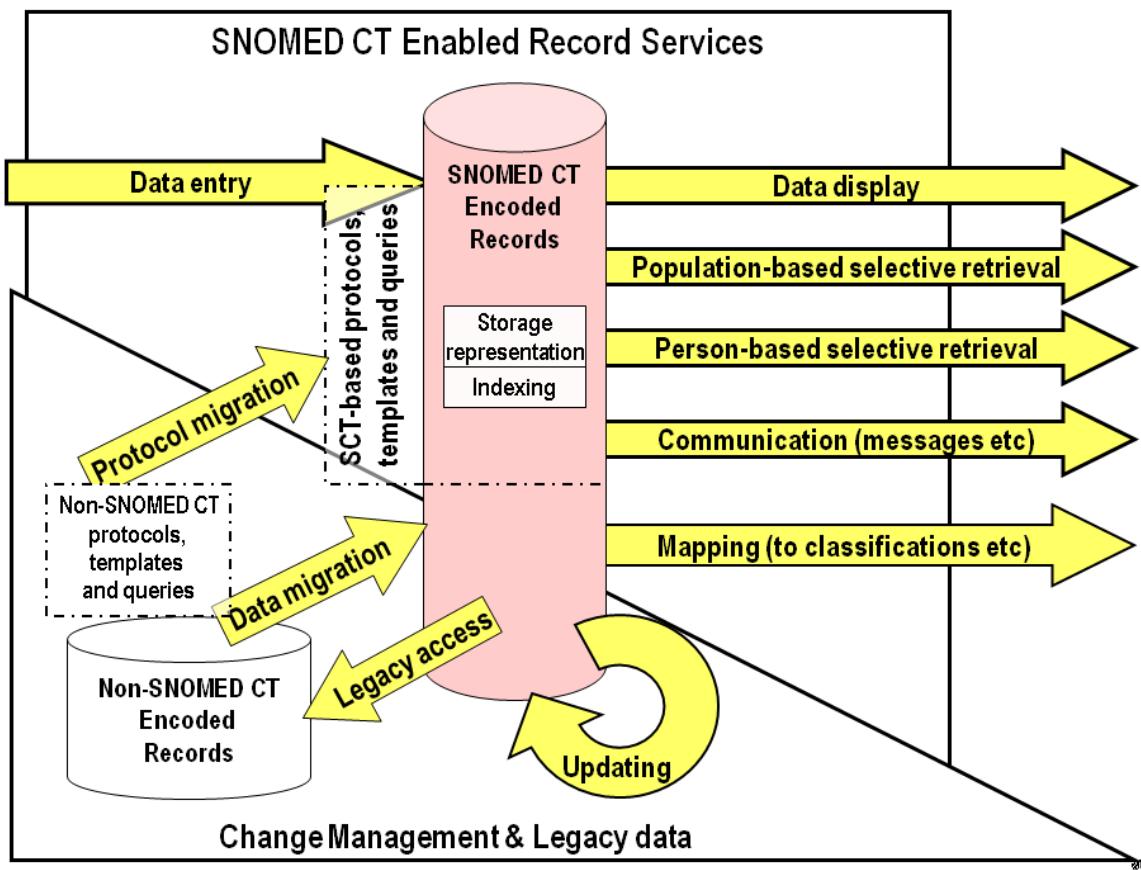


Figure 3: SNOMED CT Enabled Record Services

3.4.1 Service architecture



A *SNOMED CT enabled application* may be completely self-contained, delivering all the required services as part of a single development. Alternatively, service delivery may be modularized so that separately developed reusable modules are used to meet specific sets of requirements.

A distinction can be made between functions that only require interaction with terminology resources (*terminology services*) and functions that involve using the terminology as part of an application such as an *electronic health record* (*record services*).

Terminology services can be generalized, so that they are independent of the way the terminology is used in a particular clinical record application. *Terminology services* include support for the following types of function.

- Read-only functions:
 - Importing and updating a local terminology repository with a *SNOMED CT release*;
 - Determining the properties or an identified *component*;
 - Text or pattern searches for *Descriptions* that include a matching term;
 - Displaying a part of the *concept hierarchy*;
 - Determining whether a *SNOMED CT concept* or *expression* is equivalent to or a *subtype* of another *concept* or *expression*;
 - Locating the *cross maps* from a particular *SNOMED CT concept* to a code in another scheme or classification.
- Authoring and maintenance functions:

- Enabling the creation and maintenance of core *SNOMED CT components* to facilitate production of the *SNOMED CT International Release* and *Extensions to SNOMED CT*;
- Enabling the creation and maintenance of derivative such as *reference sets* to customize and enhance the effective use of *SNOMED CT*.

Record services are intimately related to ways in which information is entered, stored and retrieved by a particular application. Therefore, while these services interact with *terminology services* they are usually specific to a particular application or to a family of applications with a common underlying record design. *Record services* include support for the following types of function:

- *User interface* functions that:
 - Enable entry of information using *SNOMED CT expressions* where these are relevant;
 - Display of previously entered information, with appropriate rendering of *SNOMED CT expressions*;
 - Enable design of protocols that guide data entry to encourage efficient and consistent use of *SNOMED CT*;
 - Enable specification of queries that include appropriate use of *SNOMED CT* to meet requirements for selective retrieval.
- Application server functions that:
 - Store *SNOMED CT expressions* as part of the individual record entries (or in other types of instance data);
 - Communicate data including *SNOMED CT expressions* in ways dictated by standards and local specifications;
 - Apply queries to efficiently, accurately and precisely retrieve information taking account of the data structure of the application and the logical *Relationships* between *SNOMED CT expressions*.

These two sets of services can be developed and provided separately. This approach allows *record service* to access required *terminology services* through an *Application Programming Interface (API)*. The guide does not specify an *API* but, by making a clear distinction between *terminology services* and *record services*, it identifies the functions that such an interface should support.

Self-contained and modular approaches offer different profiles of advantages, some of which are summarized below.

- A modular approach offers the following advantages:
 - Rapid development of *SNOMED CT* related functionality, focused on meeting the requirements of users of a specific software application.
 - Opportunities to choose between different *terminology servers* to deliver a cost-effective solution.
 - Simplifies future migration to enhanced or more cost-effective solutions by separately identifying reusable and replaceable modules.
 - Allows several applications used by a single organization to use a single *terminology server*. This has several advantages:
 - Reduction of maintenance and support cost associated with installing each release of *SNOMED CT*;
 - Guaranteed alignment of *SNOMED CT releases* between applications that share the server;
 - Consistency of the *user interface* and technical characteristics of different applications with respect to their access to *SNOMED CT*.
- A fully integrated approach offers the following advantages:
 - Independence of third party development;
 - Customized access to *SNOMED CT* tailored to the needs of particular application users.

The approach chosen depends on a careful consideration taking into account the cost and functionality of available *components*. Commercial and technical concerns about dependence on third-party *components* may be a valid reason for in-house development of all the required services. However, even where all the development is undertaken within a single organization , separation of terminology and *record services* into separate *components* may offer a more robust approach, allowing future extensibility and migration at lower cost.

Chapter

4

4 Structure and Content Guide



This part of the guide covers the features of *SNOMED CT* that need to be understood by those implementing *SNOMED CT* in software applications. These features include the components, *derivatives* and supporting materials that are distributed as part of each *SNOMED CT Release*. In addition, the guide addresses the ways in which these components may be referenced to represent instances of clinical information in clinical records and other types of instance data.

4.1 SNOMED CT Technical Overview



This section provides an overview of the *components* and *derivatives* that form part of a *SNOMED CT release* as well as several other topics that relate to the use of *SNOMED CT* to represent instances of clinical information.

These topics are explored in more depth by other sections in this part of the guide:

- [Logical Abstract Models](#);
- [Representational Forms](#).

More detailed information about technical design and content is provided in other parts of the guide:

- [Release File Specifications \(5\)](#);
- [Concept Model Guide \(6\)](#).

4.1.1 Components



This section summarizes the essential *components* of *SNOMED CT* (*concepts*, *descriptions* and *relationships*). A *SNOMED CT enabled implementation* must be able to process and make appropriate use of these *components*, which are distributed as a set of [Release Files \(5\)](#).

4.1.1.1 Concepts



A *SNOMED CT Concept* is a clinical idea to which a unique *SNOMED CT identifier* has been assigned.

Each *Concept* is associated with:

- A unique human-readable *Fully Specified Name* (FSN), which specifies the meaning represented by the *Concept*.
- A set of other *Descriptions*, each of which represents the same *Concept* using a different human-readable *term*. These *Descriptions* support alternative representations such as *synonyms* and translations into different *languages*.
- A set of *Relationships* to other *Concepts* which provide a logical definition of the *Concept* that can be processed by a computer.

4.1.1.1.1 Concept Identifiers



Each *SNOMED CT Concept* has a permanent unique numeric *Identifier* which is known as the *Concept Identifier*.

The sequence of digits in a *Concept Identifier* does not convey any information about the meaning or nature of the *Concept*¹. The meaning of *Concept* is represented in human-readable forms by *Descriptions* and in a computer processable form by *Relationships* with other *Concepts*.

The advantages of meaningless *Identifiers* include:

- *Identifier* permanence without undermining interpretation:
 - In contrast, to maintain consistency, a meaningful code may need to change to reflect revised understanding of the nature of a disorder..
- Enabling multiple aspects of meaning to be represented in the same way:
 - A meaningful code can only represent part of meaning of a complex *concept*. For example, [staphylococcal pneumonia] is an [infection], a [respiratory disorder] and a [disorder] caused by [staphylococcus] but only one of these aspects can be represented by a code based *hierarchy*. Thus in the 'J' in the *ICD-10* code 'J152: Pneumonia due to staphylococcus' represents that fact that this is a respiratory disorder but does not represent the fact that it is an infection (codes starting with 'A') or that it is due to staphylococcus ('A490: Staphylococcal infection, unspecified').
- No artificial limitation on *concept* granularity:
 - Typical approaches to meaningful coding impose limits on both the number of levels of specificity (i.e. the length of the code) and the number of options at each level (i.e. the number of different symbols that can be used in each character position).

4.1.1.1.2 Concept granularity



The meaning represented by a *Concept* can be general (for example | procedure |), specific (for example | excisional biopsy of lymph node |) or somewhere in between (for example | biopsy of lymph node |).

- More specific *Concepts*:
 - Have finer granularity (more granular);
 - Represent clinical detail.
- More general *Concepts*:
 - Have coarser granularity (less granular);
 - Represent less clinical detail;
 - Aggregate similar *Concepts*.

Support for multiple levels of granularity allows *SNOMED CT* to be used to represent clinical data at a level of detail that is appropriate to a range of different uses.

Concepts with different levels of granularity are linked to one another by | is a | *relationships*. This enables appropriate aggregation of specific information within less detailed categories.

¹ The use of meaningless identifiers differs from the approach taken by some other coding systems and classifications. For example, the first character of an ICD-10 code indicates the general classification that it falls within.

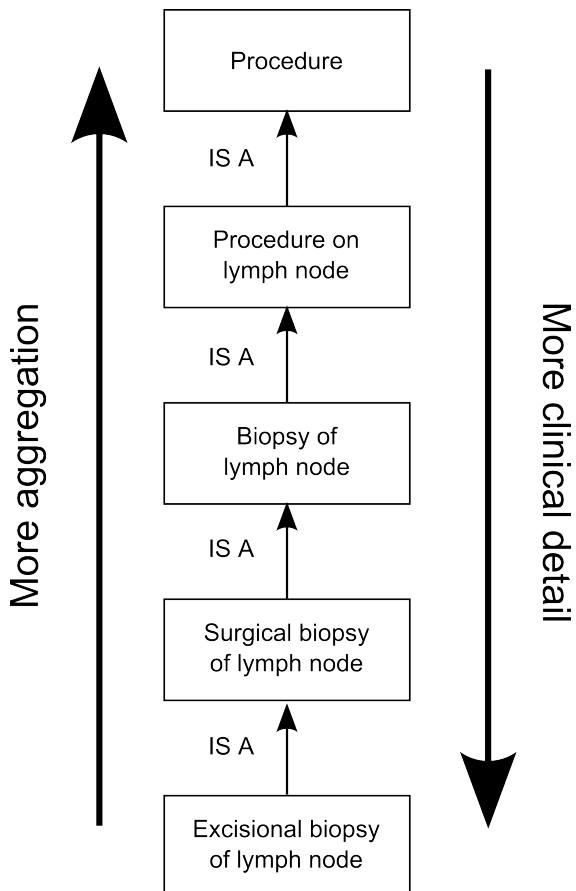


Figure 4: Multiple levels of granularity

4.1.1.2 Descriptions



A *Description* associates a human-readable *term* with a *Concept* that it describes.

A *Concept* is associated with several *Descriptions*. Each of these represents either a *Preferred Term*, *Synonym*, or *Fully Specified Name* for the *Concept* in a particular *language* or *dialect*.

A *Description* may be a *Preferred Term* in one *dialect* and a *synonym* in another *dialect*. This is indicated by references to the *Description* from an appropriate *Language Reference Set*.

Each *Description* is identified by a unique *Description Identifier* and is distributed as a row in the *Descriptions Table*.

4.1.1.2.1 Fully Specified Name



Each *concept* has one *Fully Specified Name* (FSN) intended to provide an unambiguous way to name a *concept*. The purpose of the FSN is to uniquely describe a *concept* and clarify its meaning. The FSN is not a commonly used term or natural phrase and would not be expected to appear in the human-readable representation of a clinical record.

 **Note:** The term in each FSN is unique across the entire active content of a *SNOMED CT release*.

Each FSN term ends with a “semantic tag” in parentheses. The semantic tag indicates the semantic category to which the *concept* belongs (e.g. clinical finding, disorder, procedure, organism, person, etc.). The “semantic tag” helps to disambiguate the different *concept* which may be referred to by the same commonly used word or phrase.

 **Example:** | Hematoma (morphologic abnormality) | is the FSN of the *concept* that represents the “hematoma” that a pathologist sees at the tissue level. In contrast, | Hematoma (disorder) | is the FSN

of the *concept* that represents the clinical diagnosis that a clinician makes when they decide that a person has a “hematoma”.

4.1.1.2.2 Preferred Term



Each *concept* has one *Preferred Term* in a given language *dialect*. The *Preferred Term* is a common word or phrase used by clinicians to name that *concept*.

Example: the *concept* 54987000 | repair of common bile duct (procedure) | has the *Preferred Term* | choledochoplasty | to represent a common name clinicians use to describe the procedure.

Note: Unlike the *Fully Specified Name* (FSN) the *Preferred Terms* need not be unique. Occasionally, the *Preferred Term* for one *concept* may also be a *Synonym* or the *Preferred Term* for a different *concept*. Interpretation in these cases will depend on context of use.

Example:

- | Cold sensation quality (qualifier value) | has a *preferred term* of “Cold”;
- | Common cold (disorder) | also has a *synonym* of “Cold”.

In both cases, “cold” represents a common clinical phrase used to capture the meaning of the *concept*.

Note: Selection of one term over another as “preferred” in a given language *dialect* depends entirely on whose preferences are being expressed. Different users are likely to have different preferences, and implementers are encouraged to select or create terms that properly represent the *concept* and meet the preferences of users. There is no expectation that the *Preferred Term* distributed with a given language *dialect* will meet all use cases; nor is there anything sacrosanct about the term. The U.S. English *Preferred Term* is not guaranteed to have any special status relative to other terms. Rather, it is merely one term that properly represents the *concept* and can be used as a starting point.

4.1.1.2.3 Synonym



A *synonym* represents a *term*, other than the FSN or *Preferred Term*, that can be used to represent a *concept* in a particular language or *dialect*.

Example: *Synonyms* of the *concept* 22298006 | myocardial infarction (disorder) | in English include:

- | cardiac infarction | (*Description.id*: 37442013);
- | heart attack | (*Description.id*: 37443015);
- | infarction of heart | (*Description.id*: 37441018).

The *Preferred Term* for this *concept* in English is: | myocardial infarction | (*Description.id*: 37436014).

Note: *Synonyms*, like *Preferred Terms*, are not required to be unique.

4.1.1.3 Relationships



A *Relationship* represents an association between two *Concepts*.

Each *Relationship* is identified by a unique *Relationship Id* and is distributed as a row in the *Relationships Table*.

A *Relationship* contains *Identifiers* of two logically associated *Concepts* and the *Identifier* of another *Concept* that indicates the *Relationship Type* by which they are associated.

Table 5: Example: Defining arthritis as a type of joint disorder

Relationship.id	source.id	type.id	destination.id
2227469024	3723001	116680003	399269003
In human readable <i>terms</i> ... arthritis	is a	joint disorder	

4.1.1.3.1 Relationships and concept definitions



Each *concept* in *SNOMED CT* is logically defined through its *relationships* to other *concepts*.

Every *active SNOMED CT concept* (except the *SNOMED CT Concept Root concept*) has at least one | is a | *relationship* to a supertype *concept*.

| is a | *relationships* and defining attribute *relationships* are known as the *defining characteristics* of *SNOMED CT concepts*. They are considered defining because they are used to logically represent a *concept* by establishing its *relationships* with other *concepts*. This is accomplished by establishing | Is a | *relationships* with one or more defining *concepts* (called supertypes) and modeling the difference with those supertypes through defining attributes.

👉 **Example:** | Fracture of tarsal bone (disorder) | is defined as:

- | is a | *subtype of* | Fracture of foot (disorder) |
- and has | finding site | | Bone structure of tarsus (body structure) | ;
- and has | associated morphology | | Fracture (morphologic abnormality) | .

👉 **Note:** A *relationship* is assigned only when that *relationship* is always known to be true.

👉 **Example:** Group A Streptococcus causes most cases of Streptococcal pharyngitis. However, a small percentage of these cases are caused by other species of Streptococcus. Therefore, it would be incorrect to define | Streptococcal sore throat (disorder) | as having | causative agent | | Streptococcus pyogenes (organism) |. Instead it is correctly defined as having the more general | causative agent | | Genus Streptococcus (organism) |.

4.1.1.3.2 IS A Relationships



| is a | *relationships* are also known as “Supertype - Subtype *relationships*” or “Parent - Child *relationships*”. | is a | *relationships* are the basis of *SNOMED CT*'s hierarchies, as illustrated below.

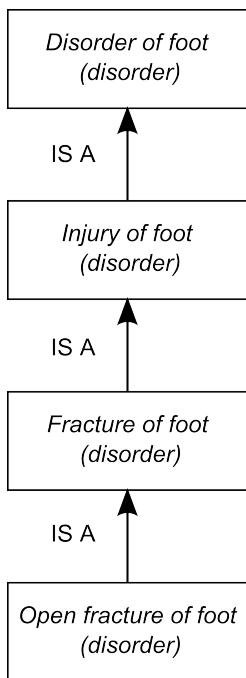


Figure 5: Example IS A hierarchy

A concept can have more than one | is a | relationship to other concepts. In that case, the concept will have parent concepts in more than one sub-hierarchy of a top-level hierarchy. Subtype relationships can be multi-hierarchical.

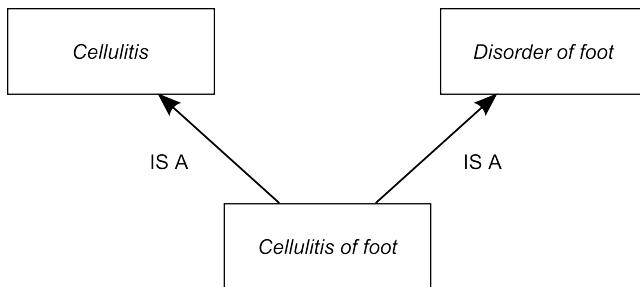


Figure 6: Example IS A Relationships

4.1.1.3.3 Attribute Relationships



An attribute Relationship is an association between two concepts that specifies a defining characteristic of one of the concepts (the source of the Relationship). Each Attribute Relationship has a name (the type of Relationship) and a value (the destination of the Relationship). For example

The combination of the attribute Relationships and | is a | relationships associated with a concept represent the logical definition of that concept. The logical concept definition includes one or more supertypes (represented by | is a | relationships), and a set of defining Attributes that differentiate it from the other concept definitions.

👉 Example:

Since pneumonia is a disorder of the lung, the logical definition of the concept | Pneumonia (disorder) | in SNOMED CT includes the following Relationship. The Attribute | Finding site | is assigned the value | Lung structure (body structure) |.

- | Finding site | = | Lung structure (body structure) |

The full definitions of the concepts | Pneumonia (disorder) |, | Infective pneumonia (disorder) | and | Bacterial pneumonia (disorder) | are shown below. Each line represents a defining Attribute with a value.

- | is a | = | pneumonitis |
- , | is a | = | lung consolidation |
- , { | associated morphology | = | inflammation |
- , | associated morphology | = | consolidation |
- , | finding site | = | lung structure | }

Figure 7: Definition of |Pneumonia (disorder)|

- | is a | = | infectious disease of lung |
- , | is a | = | pneumonia |
- , | pathological process | = | infectious process |
- , { | associated morphology | = | inflammation |
- , | associated morphology | = | consolidation |
- , | finding site | = | lung structure | }

Figure 8: Definition of |Infective pneumonia (disorder)|

- | is a | = | bacterial lower respiratory infection |
- , | is a | = | infective pneumonia |
- , | causative agent | = | bacteria |
- , | pathological process | = | infectious process |
- , { | associated morphology | = | inflammation |
- , | associated morphology | = | consolidation |
- , | finding site | = | lung structure | }

Figure 9: Definition of |Bacterial pneumonia (disorder)|

Figure 10 illustrates some of these *Relationships* graphically. | is a | Relationships relate a concept to more general concepts of the same type. In contrast, *Attribute Relationships* (such as | Finding site | and | Causative agent |) relate a concept to relevant values in other branches of the subtype hierarchy.

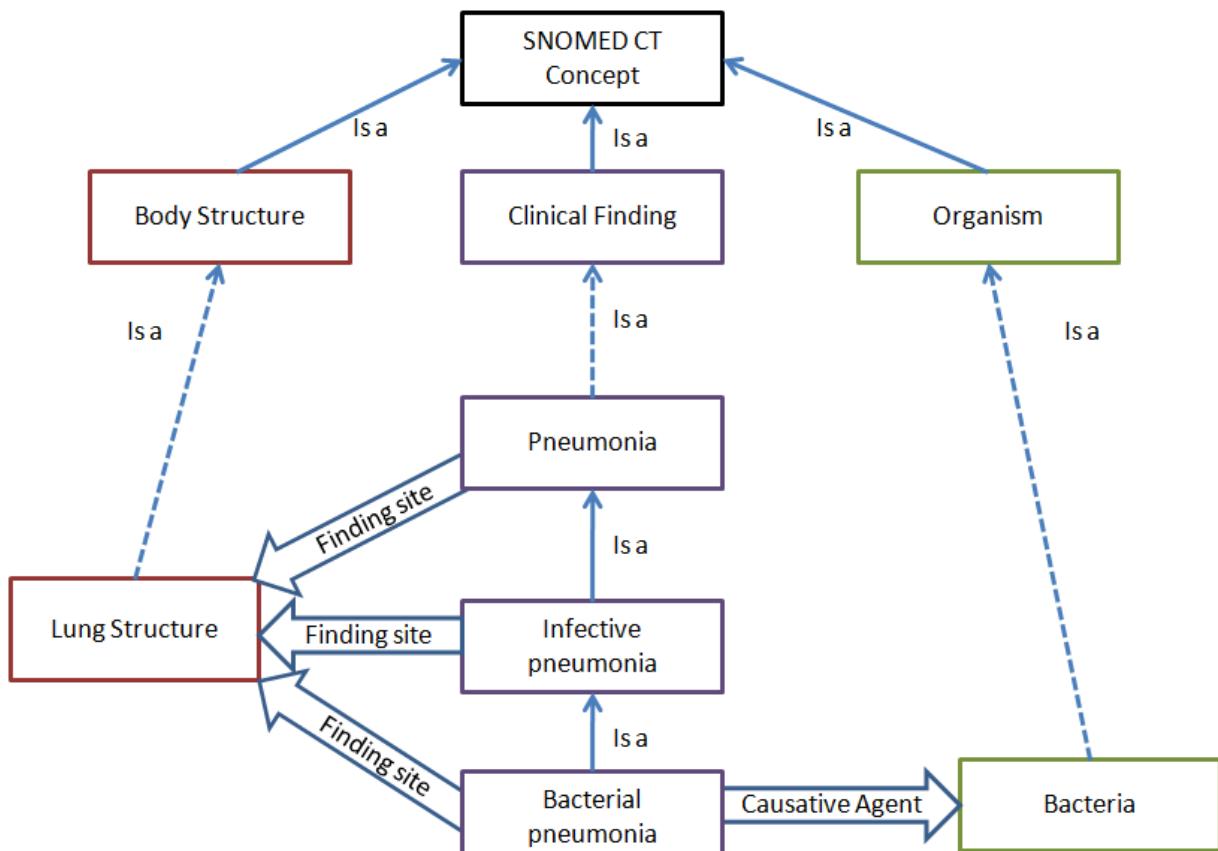


Figure 10: Illustration of Defining Relationships

4.1.1.4 Common Features of Components



This section describes common features of all *SNOMED CT Components* including identification and history management.

4.1.1.4.1 Component features - History



The content of *SNOMED CT* evolves with each release. The types of changes made include new *Concepts*, new *Descriptions*, new *Relationships* between *Concepts*, new *Cross-Maps*, and new *Reference Sets*, as well as updates and retirement of any of these *components*. Drivers of these changes include changes in understanding of health and disease processes; introduction of new drugs, investigations, therapies and procedures; and new threats to health, as well as proposals and work provided by *SNOMED CT* users.

Once released, the unique *Identifiers* of *SNOMED CT components* are persistent, and their *Identifiers* are not reused. *Concepts* and *Descriptions* continue to be distributed even when they are no longer recommended for active use. This allows a *current* release to be used to interpret data entered using an earlier release.

Since the implementation of *Release Format 2* (RF2), all changes in components are represented in the corresponding files, by adding a new row, with the same component ID, a new effective time and any necessary change in the component values.

The Component Inactivation *Reference Sets* are used to indicate the *reason* for inactivating a *component*. These *reasons* include errors, duplication of another component and ambiguity of meaning, and the files are used to describe reasons for inactivation of *Concepts*, *Descriptions* and *Relationships*. Some *SNOMED CT Concepts* represent classification *concepts* that have imprecise and potentially changeable meanings. These

are marked with the inactivation indicator value [900000000000486000] and were considered *active* until the January 2010 release of *SNOMED CT*. All Limited *concepts* are now considered to be *inactive*.²

4.1.1.4.2 Component features - Identifiers



Components within *SNOMED Clinical Terms* are identified and referenced using numeric *Identifiers*. These *Identifiers* have the data type *SCTID* (*SNOMED CT Identifier*).

The *SCTID* data type is 64-bit *integer* which is allocated and represented in accordance with a set of rules. These rules enable each *Identifier* to refer unambiguously to a unique component. They also support separate partitions for allocation of *Identifiers* for particular types of component and *namespaces* that distinguish between different issuing organizations.

4.1.2 Derivatives



This section describes *derivatives* that are specified by and distributed as part of *SNOMED CT*. *Derivatives* are artifacts which are either required or useful to support some aspect of *SNOMED CT enabled implementation*. These artifacts are known as *derivatives* because they are derived from *SNOMED CT Components* and either add properties to them or specify sets of related *components*. All *SNOMED CT enabled applications* need to support some *derivatives*.

The set of *derivatives* that need to be supported by an implementation depend on user requirements for particular types of functionality. Important aspects of functionality that require support for relevant *derivatives* include:

- Tracking changes to the status of *components*;
- Filtering and prioritizing searches;
- Representing alternative *navigation hierarchies*;
- Adding annotations to *components*;
- Cross-Mapping to and from other coding schemes and classifications.

4.1.2.1 Reference Sets



Reference Sets represent groups of *components* that share specified characteristics that affect the way the components are displayed or otherwise accessible within a particular *realm*, specialty, application or context.

Different types of *Reference Set* are used to represent:

- *Language* and *dialect* variations in the use of particular *terms* to describe a *Concept*;
- Subsets of component that are included in or excluded from the set of values that can be used in a particular country, organization , specialty or context;
- Frequency of use of *Descriptions* or *Concepts* in particular country, organization , specialty or context;
- Suitability of particular *Concepts* for use in a particular context in a record or message;
- Structure and ordering of hierarchies displaying *Concepts* for user *navigation*.

Reference Sets can be represented using the *Subset* and *SubsetMembers* files of *Release Format 1* or using the *Refsets* files specified by *Release Format 2*. In both cases, each rows in these tables represents a *component* that is a member of the set and may associate some additional information with the referenced *component*.

Some types of *Reference Set* may also be represented by a set of rules referred to as an 'intensional *Refset* definition'.

² Some Concepts derived from classifications such as ICD-10 include the abbreviations NOS (not otherwise specified) or NEC (not elsewhere classified). These are only valid in respect of a particular classification and change in their meaning if additional precisely defined codes are added to that part of the classification. Furthermore, a Concept that is not otherwise specified in ICD-10 may well be more precisely represented by another SNOMED CT Concept and thus from a SNOMED CT perspective "otherwise classified."

4.1.2.2 Navigation Hierarchies



SNOMED CT subtype Relationships provide a logical semantic *hierarchy*. Often it is possible to view parts of the terminology and select particular *Concepts* by navigating through this *subtype hierarchy*. However, there are many situations in which the pure *subtype hierarchy* does not provide an ideal route for navigating the *hierarchy*.

Navigation links are used to provide an alternative route through parts of the terminology. A *navigation link* can link any two *Concepts* together to identify a useful route for *navigation*. Each of the *navigation links* is directional, linking a navigational parent *Concept* to a more refined navigational child *Concept*. However, unlike the *subtype relationship* the presence or absence of a *navigation link* neither adds to nor subtracts from the definition of either of the *Concepts* that it links.

Some *Concepts* may exist only to provide nodes in a *navigation hierarchy*. These *Concepts* are subtypes of |*Navigational Concept*| and play no part in the semantic definitions of any other *Concept*.

4.1.2.2.1 Uses of Navigational Hierarchies

4.1.2.2.1.1 Breaking down a subtype into manageable categories



Some *Concepts* have a large number of *subtype children* that cannot be logically divided into intermediate *subtypes*. At the *user interface* these result in long lists of options, which are difficult to visualize and navigate. Navigational *Concepts* with appropriate navigational links to the *supertype parent* and its *subtype children* provide an intermediate layer without disrupting the semantic definitions.

The |*clinical finding*| top-level *Concept* has a large number of *subtype children*. Intermediate *navigation Concepts* group some of these together in a convenient way.

👉 Example:

Three *subtypes* related to pregnancy are grouped together under a single natural navigational *Concept*:

- Disorder of pregnancy / labor / delivery / puerperium [*navigation concept*];
- Disorder of pregnancy;
- Disorder of labor / delivery;
- Disorder of puerperium.

4.1.2.2.1.2 Bypassing levels in the subtype hierarchy



Some *Concepts* that are members of the same rational set of choices may be found at different levels in the *subtype hierarchy*. This may occur because some have intervening *subtypes* and some of these intervening *concepts* may not be required for data entry. Addition of new *concepts* in a release may change the *concepts* available at some levels in the *subtype hierarchy*. *Navigation links* can "bypass" levels in the *subtype hierarchy* to represent a rational sets of choices for use in a particular situation.

👉 Example:

While it is semantically correct to nest | common cold | in the following *subtype hierarchy*, a user may reasonably expect to see "common cold" as an immediate navigational child of | upper respiratory infection |.

- | upper respiratory infection |
 - | Viral upper respiratory tract infection |
 - | common cold |

4.1.2.2.1.3 Linking related Concepts of different types



Navigational links can also be used to provide access to connected *Concepts* even when they are from different *hierarchy branches*.

Example:

A *navigation links* could associate:

- "hypertension" (the disorder) with | blood pressure | (the observation);
- | cataract | (disorder / finding) with "cataract surgery" (the procedure).

4.1.2.2.1.4 Ordering the display of subtypes



Sibling Concepts in a *subtype hierarchy* are not ordered. However, at the *user interface* a particular *order* may be useful to highlight commonly used Concepts or to mirror a conventional ordering.

Example:

Vertebrae, cranial nerves, disease stages, etc.

Navigational links are ordered and are used to impose *order*, even when the set of navigational *children* is the same as the set of *subtype children*.

4.1.2.2.1.5 Providing alternative hierarchies



The *subtype hierarchy* is logically defined and there can only be one such *hierarchy*. However, as *navigation hierarchies* have no definitional consequences, it is possible to have different hierarchies for different groups of users with differing needs.

Initial releases of *SNOMED CT* will contain a single set of *navigation links* but those engaged in technical implementation should be aware that in the future there may be separate sets of *navigation links* for use in different environments.

4.1.2.3 Cross~Maps



SNOMED CT specifications and content include resources that support *Cross-Mapping* to and from other code systems, classifications and terminologies. These resources support simple mapping, where there is a one-to-one *Relationship* between a *SNOMED CT concept* and code in a *target scheme*, and more complex maps where these are required.

More complex mapping requirements supported by the *SNOMED CT Cross-Mapping* model include:

- Maps from a single *SNOMED CT concept* to a combination of codes (rather than a single code) in the *target scheme*.
- Maps from a single *SNOMED CT concept* to choice of codes in the *target scheme*. In this case, the resolution of the choices may involve:
 - Manual selection supported by advisory notes.
 - Automated selection based on rules that test other relevant characteristics in the source data (e.g. age and sex of the subject, presence or absence of co-existing conditions, etc).
 - A combination of automated processing with manual confirmation or selection where rules are insufficient to make the necessary decisions.

In *Release Format 2* *Cross~Maps* are represented using *Reference Sets*. The type of *Reference Set* used varies according to the nature and complexity of the mapping, there is a *Simple Map Reference Set* and a *Complex Map Reference Set*.

4.1.2.4 Search support



The *Developer Toolkit*, which is supplied as part of the *SNOMED CT International Release*, includes several tables that can be used to simplify and support for text searching.

There are two *WordKey Tables*. These tables link each word used in *SNOMED CT* to every:

- *Description* in which it is used;
- *Concept* associated with an *active description* in which the word is used.

There are also two *Dualkey Tables*. These tables link each abbreviated word pair to every:

- *Description* in which that pair of words is used;
- *Concept* in which the combined set of *active descriptions* contains that pair of words.

These tables are provided to assist implementation. However, use of these tables is optional, as developers may generate and use alternative search support resources.

An extended version of the *Developer Toolkit*, provides Java® programs to generate indexes that may be useful to organizations that develop *SNOMED CT Extensions*.

4.1.3 Extensions



SNOMED CT is designed to allow the *International Edition* to be enhanced by adding *Extensions* that meet national or local requirements. *Extensions* are managed by *IHTSDO Members* or *Affiliates* who have been issued with a *Namespace Identifier*, which distinguishes the *Identifiers* of the *Components* they maintain. An *Extension* may contain *Components* of various types (e.g. *Concepts*, *Descriptions*, *Relationships*, and *Derivatives* including *Reference Sets* used for a variety of purposes).

4.1.3.1 Rationale for Extensions



SNOMED CT is a detailed clinical terminology which covers a broad scope. However, some groups of users will need additional *Concepts*, *Descriptions* or *Reference Sets* to support national, local or organizational needs.

This section explains the structures that enable *IHTSDO Members* (*National Release Centers*) and *IHTSDO Affiliates* to add *Concepts*, *Descriptions*, *Relationships* and *Reference Sets* to complement the *SNOMED CT International Release*.

The *Extension* mechanism allows *SNOMED CT* to be adapted to address the terminology needs of a country or organization which are not met by the *International Release*. The mechanism provides a structure within which the components of each *Extension* are uniquely identified and attributed to a specific issuing organization. This ensures that, when instance data containing content from different *Extensions* if communicated, the provenance of each referenced *Concepts* is clear and ambiguity is avoided. Since the *International Release* and all *Extensions* share the same common structure, the same application software can be used to enter, store and process information from different extensions. Similarly, *Reference Sets* can be constructed that refer to content from the *International Release* and a variety of *Extensions*.

The common structure also means that, content developed by one organization can where relevant be easily submitted for possible inclusion in a *National Edition* or in the *International Edition*.

Using the *extension* structure can also help organizations transfer responsibility for terminology to the *IHTSDO* or to another organization , subject to the *terms* of the *Affiliate License* .

- Local content requirements that are likely to have wider applicability should be submitted to a *National Release Center* for consideration.
- National requirements likely to have International value should be submitted to the *IHTSDO* so they can be considered for inclusion in the *International Edition*.

4.1.3.2 Practical uses of Extensions



An *Extension* mechanism offers many advantages to developers, vendors, terminologists, national bodies and users.

Such a mechanism allows:

- **Users** to access the *SNOMED CT International Release* and one or more *Extensions* through a single *user interface*;
- **Developers** to implement *SNOMED CT Extensions* without developing specialized software;
- **Vendors** to develop and sell products to take advantage of both *International Release* content and *Extensions*;

- **Organizations** to develop and share terminology that meet their business needs, without procuring software;
- **IHTSDO Affiliates** to develop terminology that can be shared with other organizations and considered for addition to the *International Release* content;
- **IHTSDO Affiliates** to use locally-developed terminology without potential overlap with the work of other organizations .

This structure also enables specialized *Concepts* and *Descriptions* within an *Extension* to be related to *Concepts* and *Descriptions* distributed as part of *SNOMED CT*.

- An *Extension Concept* may be:
 - A national or organizational definition of a *concept*, which is more rigorous or specific than that generally applied to the *SNOMED CT Concept*;
 - An experimental procedure that is not established sufficiently to merit the inclusion in the main body of *SNOMED CT* but which may be in a local controlled study.
- *Extension Descriptions* may be colloquial *synonyms* for a *SNOMED CT Concept* or *descriptions* for an *Extension Concept*.
- *Extension Relationships* may be required to allow analysis packages or decision-support protocols to access additional information about a *SNOMED CT Concept* or to describe *relationships* between *Extension Concepts*:
 - Links between local procedures and relevant administrative actions;
 - Links between local procedures and *SNOMED CT Procedures*.
- *Extension Reference sets* may group *SNOMED CT Concepts* in ways that are specific to data entry contexts of a particular application or communication specification.

The *Concepts*, *Descriptions*, *Relationship* and *Reference Sets* that form an *Extension* must be:

- Distinguishable from the main body of *SNOMED CT*, not only in the thesaurus, but also when stored in a patient record, *query* or decision support protocol;
- Distinguishable from other *Extensions*, in the same way as they are distinguishable from the main body of *SNOMED CT*;
- Able to be distributed and processed in the same way as equivalent *components* from the main body of *SNOMED CT* without requiring specific adaptations of *SNOMED-enabled applications*.

The requirements for *Extensions* can be summarized as follows:

- Support for extra terminology *components* including *Concepts*, *Descriptions*, *Relationships* and *Reference Sets*:
 - These extra *components* should behave as though they were *components* of *SNOMED CT* but they should be distinguishable from *components* that are part of the *SNOMED CT International Release*.
- Globally unique identification of any terminology *component* that may be used outside the scope of a limited local environment:
 - The mechanism must allow several organizations to issue mutually exclusive *Identifiers* for *components* of their *Extensions*.
 - To avoid the risk of misinterpretation, this mechanism must be effective in various contexts including:
 - Within the thesaurus;
 - In patient records;
 - In queries, decision-support protocols or knowledge bases.
 - The mechanism must indicate when *Concepts* have moved, or are expected to move, between an *Extension* and the *International Release*, or from one *Extension* to another.

- A shared understanding of the responsibility of an organization that creates an *Extension* and provides it for the use of other organizations . These responsibilities include:
 - Maintenance of the *Concept*, *Descriptions*, *Relationships*, and *Reference Sets*;
 - Inactivation of these *components* as appropriate (duplication, ambiguous, outdated, etc.);
 - Submission to an *IHTSDO Member's National Release Centre* for consideration as an addition to a *National Edition* or to the *International Release* content.

4.1.4 Instance data



4.1.4.1 Introduction



This section describes the use of *SNOMED CT* to express clinical ideas in patient records, messages, documents, decision support protocols, queries and other artifacts .

Applications need to create, manipulate and consistently interpret standard *SNOMED CT* representations in instance data to support the entry, storage, retrieval and communication of clinical information.

4.1.4.2 Expressions



An *expression* is a structured combination of one or more *concept identifiers* used to express an instance of a clinical idea.

- ***precoordinated expression:*** An *expression* containing a single *concept identifier* is *precoordinated*. The clinical idea it expresses is represented by the identified *concept*. The defining *relationships* of that *concept* precoordinate its meaning.
- ***postcoordinated expression:*** *expression* that contains two or more *concept identifiers* is *postcoordinated*. The *concept identifiers* in a *postcoordinated expression* are related to one another in ways that build a more specific clinical idea. The required meaning is expressed by postcoordinating several clinical ideas each of which is represented by an identified *concept*.

👉 **Example:** A *postcoordinated expression* can indicate the specific site of a finding even when that specific combination of disorder and site is not represented by a single *SNOMED CT Concept*.

4.1.4.3 Terminology Bindings



Terminology binding is one part of the process of specifying *constraints* on the way that information is structured and represented.

Consistent representation is a prerequisite for effective retrieval and reuse of clinical record information. Requirements for reuse are many and varied, ranging from direct support for the care of the individual patient, through to aggregate analysis for research, statistics and audit. The common theme of these requirements is the need to retrieve particular items of information reliably and consistently, irrespective of the environment in which the data was entered and stored.

Since both the information model and *SNOMED CT* contribute to the processable meaning of an entry in a clinical record it is essential to manage the interdependencies between these two components.

Simple requirements can be addressed by specifying a value-set consisting of the permitted coded values that can be used in a particular field. However, effective representation of clinical records requires a rich information model and an expressive terminology.

Models such as *EN13606* and the *HL7 RIM* provide the necessary structural flexibility and *SNOMED CT postcoordinated expressions* provide expressivity. An inevitable side-effect of a richer approach to information representation is an increase in the interdependencies and overlaps between the information model and the terminology. In order to specify and validate consistent representation of meaningful clinical records, *constraints* must be applied to both the information model and terminology. These *constraints* must address all the facets of the model and terminology (e.g. including the use of *postcoordination* and the effect of modeled record

structures). The *constraints* on information model and terminology components must be integrated, or bound together, in ways that ensure consistency, avoid ambiguity and minimize the number of different ways in which the same meaning may be expressed.

A terminology binding is an instance of a link between a *terminology component* and an *information model artifact*. Therefore, the document considers the representation of the required *terminology components* and the way these are associated with relevant *information model artifacts*.

The *information model artifact* to which a *terminology binding* is applied may be a field of a class in a static model or a collection of *fields* of one or more related classes.

Bound components include:

- Information model artifacts :
 - Coded attributes in an *HL7 Version 3* model, an *EN13606 Archetype* or in the proprietary information model of an operational application.
- Terminology components:
 - *Constraints on SNOMED CT expressions*.

4.1.4.4 Expression Constraints



SNOMED CT contains hundreds of thousands of *Concepts* and this rich resource is greatly expanded by use of *postcoordinated expressions*. In any given situation the range of *Concepts* or *expressions* that are useful, relevant and meaningful is much more limited. This gives rise to a requirement to represent *constraints* on the content or a particular field in a way that can be interpreted and applied by application software.

The simplest *constraint* requirements can be met by specifying the list of valid codes. This requirement is addressed by *subsets* specified using the *Reference Set* mechanism. In some cases, it is useful to express the range of possible values 'intensionally' by specifying rules rather than by listing every member of the set (e.g. to include all *concepts* that are *subtypes* of a specified *concept*).

The use of *postcoordinated expressions* adds further dimensions to the requirement for *constraints*. It may be necessary to specify whether all *postcoordinated refinements* of *concept* are permitted or whether some types of *refinement* are prohibited or required. It may also be necessary to specify whether a *postcoordinated expression* that is equivalent to a permitted value is itself permitted.

Requirements for representing *expression constraints* are closely related to the requirements for representing *query predicates* in queries.

4.1.4.5 Query Predicates



Queries to be applied to *electronic health records* that including *SNOMED CT expressions* may need to represent predicates that test *postcoordinated expressions*. The requirements for representing *postcoordinated expression query predicates* are closely related to the requirements for representing *constraints* on *expressions*. While a *constraint* specifies whether a particular *expression* is permitted in a particular situation, an *expression predicate* specifies the range of candidate *expressions* that match the *query*.

4.2 Logical Abstract Models



This section provides a logical abstract view of *SNOMED CT components* and *derivatives*; and the use of these to represent instances of clinical information. Subsequent sections provide detailed technical *descriptions* of the *SNOMED CT components*, *derivatives* and related artifacts .

4.2.1 Logical Model of SNOMED CT Components



The abstract logical model of *SNOMED CT components* is illustrated by [Figure 11](#). The model is centered around the representation of *concepts* and their associated *relationships* and *descriptions*.

Alignment between *release files* and the logical model:

- *SNOMED CT Release Format 2* is closely aligned with the logical model;
- *A mapping table* is provided with the *Release Format 1* file specification to map *RF1* file structures to the abstract model.

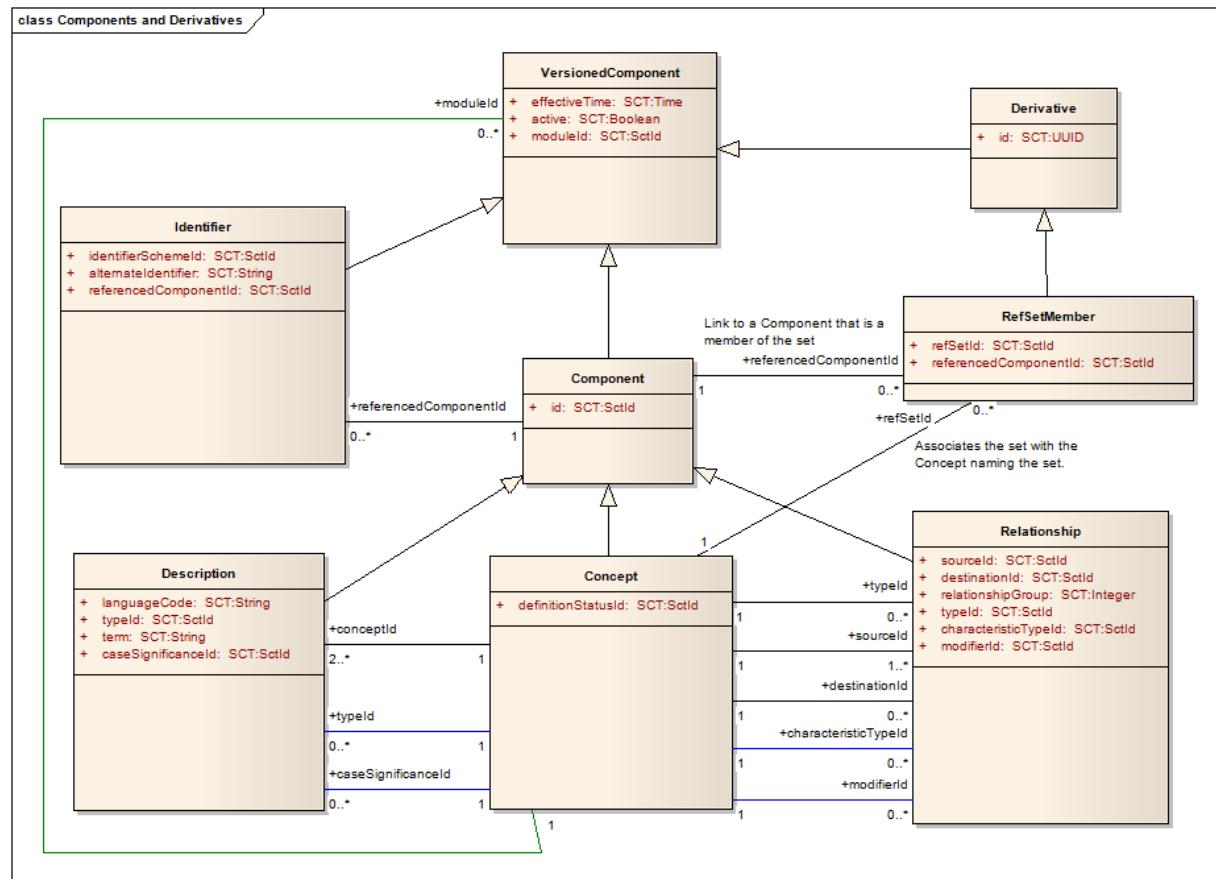


Figure 11: Abstract logical model of SNOMED CT components

4.2.1.1 Descriptions and other non-definitional properties of concepts

4.2.1.1.1 Descriptions

The set of *terms* that describe a *concept*. These include *fully specified names*, *preferred terms* and *synonyms* in each supported *language*.

4.2.1.1.2 Legacy codes

Legacy *Identifiers* present in the *SNOMED CT* logical model include the *CTV3ID* (the *Read Code* from *NHS Clinical Terms Version 3*) and the *SnomedId* (the *SNOMED* code used in *SNOMED 3*).

👉 **Note:** The *CTV3ID* and *SNOMEDID* fields are no longer supported in *Release Format 2*. Instead a *|Simple map (reference set)* is used to document the link between legacy codes and *SNOMED CT*.

👉 **Note:** The *CTV3ID* field should no longer be relied upon for mapping to and from the *Read Codes*. Additional mapping work in the UK identified some anomalies and resulted development of more flexibility table for *Read Code Mapping*



4.2.1.1.3 Cross~!Maps



Cross~!Maps to other terminologies and classifications are indirectly a part of the logical *SNOMED CT* model for *concepts*. However, this aspect of the model is outside the scope of this guide.

4.2.1.2 Relationships and concept definitions



Each *concept* is defined by a set of *relationships* to other *concept*. The resulting definition may be sufficient to distinguish the *concept* from its parents and siblings in the *subtype hierarchy* in which case the *concept* is considered to be *fully defined*. If the definition is not sufficient to distinguish the *concept* from its parents and siblings, the *concept* is said to be *primitive*. The *concept* contains a field that is set to indicate whether its definition status is *primitive* or *fully defined*.

Figure 12 illustrates the abstract logical model of a *concept*, including the defining *Relationships* between *concepts* (represented by the associations labelled *sourceld*, *destinationId* and *typeId*) and the definition status (represented by the *definitionStatusId*).

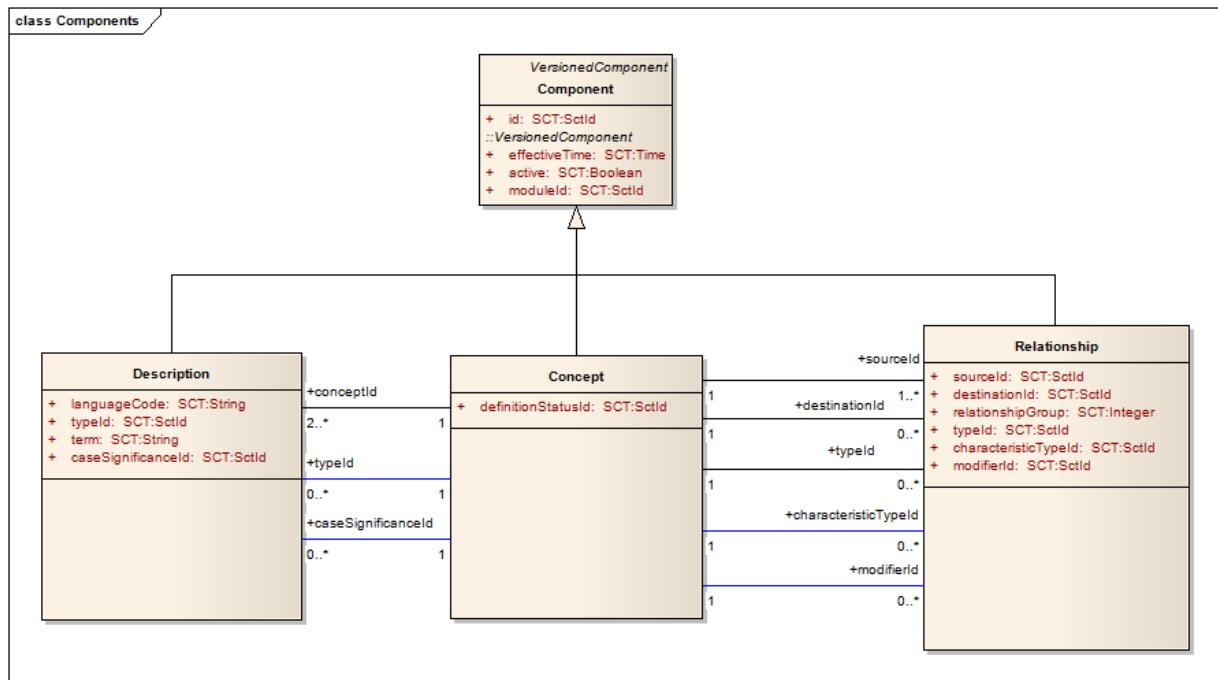


Figure 12: General Abstract Logical Model of a SNOMED CT concept definition

4.2.1.3 Alternative logical abstract model views of concept definitions



The definition of a *concept* can be logically transformed between different views without loss of meaning based on the definitions of related *concepts*.

For example:

Consider the following set of defining *relationships*:

```

| pain in upper limb || is a || pain |
| pain in upper limb | has | finding site || upper limb structure |
| hand structure || is a || upper limb structure |
| Hand pain || is a || pain |
| Hand pain | has | finding site || hand structure |

```

Based on the above five *relationships* it is possible to infer a new *relationship*:

```
| Hand pain || is a || pain in upper limb |
```

The definition of | Hand pain | can thus be viewed in three semantically identical forms:

1. As originally stated :

- | Hand pain | | is a | | pain | and has | finding site | | hand structure |

or

2. With the additional inferred *relationship*:

- | Hand pain | | is a | | pain | and | is a | | pain in upper limb | and has | finding site | | hand structure |

or

3. With the inferred *relationship* but without the redundant stated *relationship* | is a | | pain | : :

- | Hand pain | | is a | | pain in upper limb | and has | finding site | | hand structure |

The *relationship* | is a | | pain | is redundant because this can be determined by traversing the | is a | *relationship* to | pain in upper limb | which in turn is defined as | is a | | pain |.

The result of manipulations like this is that several distinct views of the logical abstract model can be described based on the manner in which they are derived.

Different views of *concept* definitions vary in one or more of the following three dimensions:

- Flattened or nested;
- Stated or inferred;
- Direction and extent of logical *transformation*

These three dimensions are considered in the following subsections of this guide.

4.2.1.3.1 Flat and nested definition views

4.2.1.3.1.1 Flat definition views



In a flat view a *concept definition* consists only of defining *relationships* with target values that are themselves identified *concepts*.

To support this view *concepts* must be created (and defined) for any value that needs to be expressed in the definition of another *concept*.

👉 Example: The | finding site | for the concept | pain in left hand | could only be defined by first creating a concept | structure of left hand | leading to a definition such as:

| pain in left hand | has | is a | | pain |.
| pain in left hand | has | finding site | | structure of left hand |.

The concept | structure of left hand | could be defined as follows:

| structure of left hand | | is a | | hand structure |
| structure of left hand | has | laterality | | left |.

4.2.1.3.1.2 Nested definition views



In a nested view of a *concept definition* the target value of a defining *relationship* may itself be a nested definition.

This avoids the need for creating intermediate *concepts* but results in more complex definitions.

👉 Example:

The | finding site | for the concept | pain in left hand | could be defined without creating the concept | structure of left hand | by nesting an appropriate definition as follows:

| pain in left hand | | is a | | pain |
| pain in left hand | has | finding site | (| is a | | hand structure | and has | laterality | | left |).

4.2.1.3.1.3 SNOMED CT support for flat and nested definition views



Currently the *SNOMED CT* editing environment works with flat definition views and the standard relational distribution files do not support nested definition views.

Views of *concept definitions* that include nested definitions can be generated from existing *SNOMED CT* data. The proposed *SNOMED CT* XML distribution format does have the potential to support nested views.

Logically the flat form is as expressive as the nested form. The only difference is the need to create and define *concepts* to represent the nested elements in the definition.

Example:

To allow the *concept* | pain in left hand | to be *fully defined* without using a nested definition, | structure of left hand | must exist as a *concept* in *SNOMED CT*.

4.2.1.3.2 Stated and inferred definition views

4.2.1.3.2.1 Stated definition view



A stated *concept definition* is the set of *relationships* (and groups of *relationships*) that an author (*modeler*) has stated to be *defining characteristics* of a *concept*. The *stated view* is maintained in the *SNOMED CT* editing environment and is reviewed and modified during the process of editing a revised edition of *SNOMED CT*.

The *stated view* is distributed in a format similar to the *relationship file*.

4.2.1.3.2.2 Inferred definition views



Inferred *concept definitions* are derived from a stated *concept definition* taking account of the definitions of the *concepts* referred to in the stated definition.

Inferences are derived by applying a consistent set of logical rules to the definition taking account of the definitions of related *concepts*.

Several semantically identical views may be inferred and these are discussed in the following section.

The standard *SNOMED CT* distribution includes the *relationships table* which represents one of the inferred views of the definitions of all *active concepts*.

4.2.1.3.2.3 Alternative inferred definition views



Several semantically identical views may be inferred by applying different logical *transformations* to the *stated view*. Logical *transformations* may vary in the extent to which they normalize the definition and the level of redundancy in the resulting definition.

Different inferred views have properties that optimize different types of function.

The extreme points in the spectrum of possible *concept definition* views are:

- Comprehensive:
 - The set of all defining *relationships* that can be inferred to be true for a *concept* based on the stated definition of this *concept* and the stated definitions of all other directly or indirectly related *concepts*.
- Minimal:
 - The smallest set of defining *relationships* that expresses the definition of the *concept*.

Each inferred view is a combination of a specific *supertype view* (| is a | *relationships*) and an *attribute view* (other defining *relationships*).

4.2.1.3.2.3.1 Supertype aspects of inferred definition views



An inferred definition view includes one of several alternative views of the supertype | is a | *Relationships*. The considerations in this section exclude the *defining characteristics* of a *concept*.

4.2.1.3.2.3.1.1 Comprehensive view of supertype ancestors ("transitive closure")



An inferred *concept definition* view may explicitly contain *relationships* to all supertypes *ancestors* of the defined *concept*.

This comprehensive view of supertypes is known in *description logic* as a "transitive closure". It involves traversing (transiting) the target of each *is a* *relationship* to look for and follow further *is a* *relationships* until all paths through the *hierarchy* reach the *root concept* (closure).

This is a highly redundant *expression* of the logical abstract model of a *concept definition*. Applied to the full content of *SNOMED CT* it results in tens of millions of *relationships*.

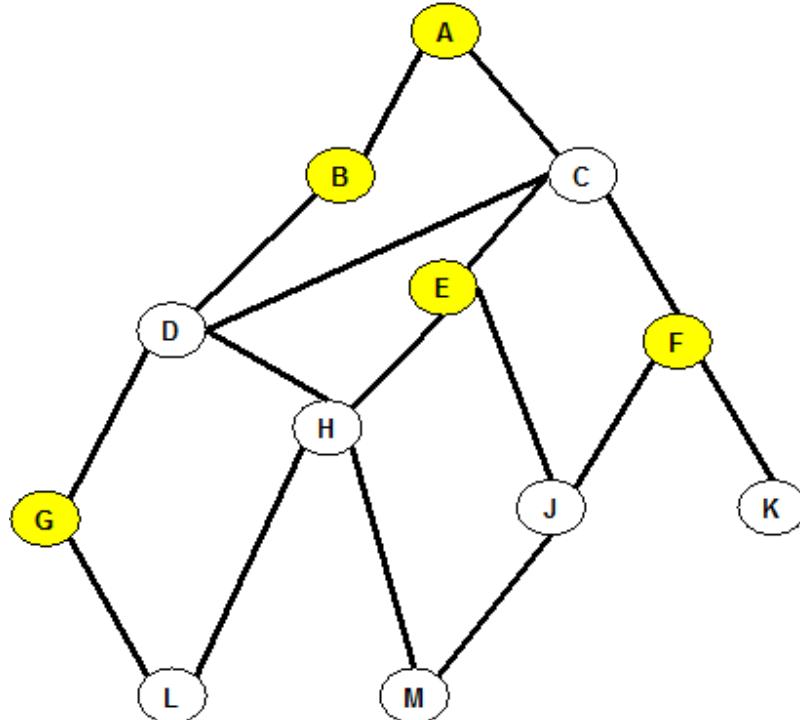


Figure 13: Example hierarchy with list of supertypes in the transitive closure

Table 6: Transitive Closure of Supertypes in the Example Hierarchy

Concept	Transitive closure of supertypes
A	-
B	A
C	A
D	A, B, C
E	A, C
F	A, C
G	A, B, C, D

Concept	Transitive closure of supertypes
H	A, B, C, D, E
J	A, C, E, F
K	A, C, F
L	A, B, C, D, E, G, H
M	A, B, C, D, E, F, H, J

The advantage of this type of view is that there is no need to walk the *hierarchy tree* to answer the question "is concept M subsumed by concept B". Instead this can be answered simply by checking the *transitive closure* of "concept M" for the presence of "concept B". Therefore, this view enables high-performance subsumption testing.

 **Note:** Experience suggests that a pre-computed *transitive closure* table out-performs other options and is robust, flexible and easy to implement. Therefore, unless storage capacity is significant concern, this approach is recommended.

4.2.1.3.2.3.1.2 Proximal supertype view (standard distribution view)



An inferred view of a *concept definition* may contain *relationships* to the set of proximate *supertype parents* of that *concept*. *Relationships* with other *supertype ancestors* that can be reached by traversing multiple | is a | *relationships* are omitted.

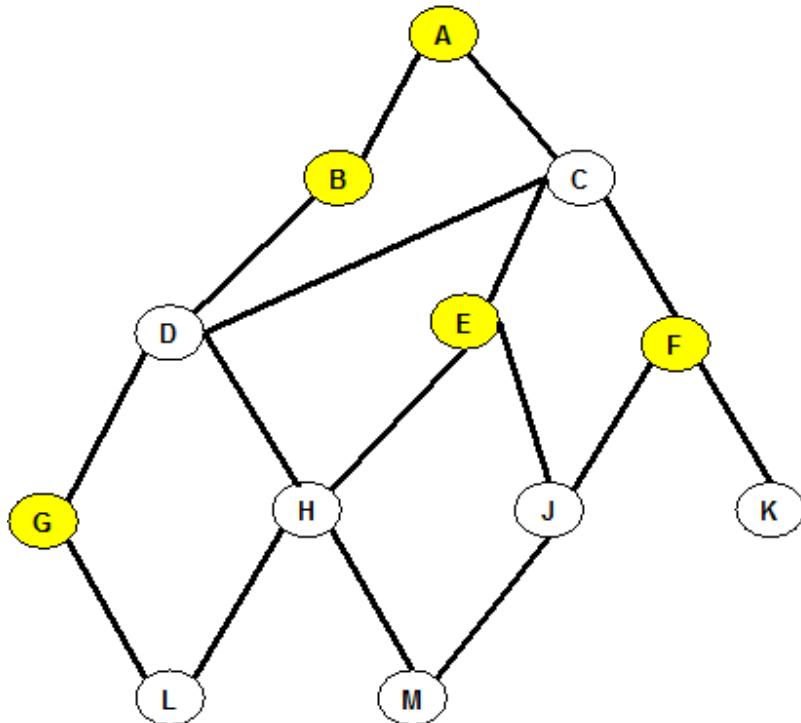


Figure 14: Example hierarchy with list of proximal supertypes

Table 7: Proximal Supertypes in the Example Hierarchy

Concept	List of proximal supertypes
A	-
B	A
C	A
D	B, C
E	C
F	C
G	D
H	D, E
J	E, F
K	F
L	G, H
M	H, J

4.2.1.3.2.3.1.3 Comprehensive primitive supertype view



An inferred view of a *concept definition* may contain *relationships* to all *supertype ancestors* that are "*primitive*" *concepts* (yellow shaded in examples).

The rationale for this is that all the distinguishing features of the "*fully defined*" *concepts* (white unshaded in examples) are represented by other defining *relationships* which will show up in the attribute part of the view.

This view can be used when testing whether a candidate *concept* is subsumed by a predicate *expression*. If the proximal *primitive* supertype view of the predicate *expression* includes *any concept* that is not in the comprehensive *primitive* view of the candidate *concept definition*, then the *concept* is not subsumed by the *expression*.

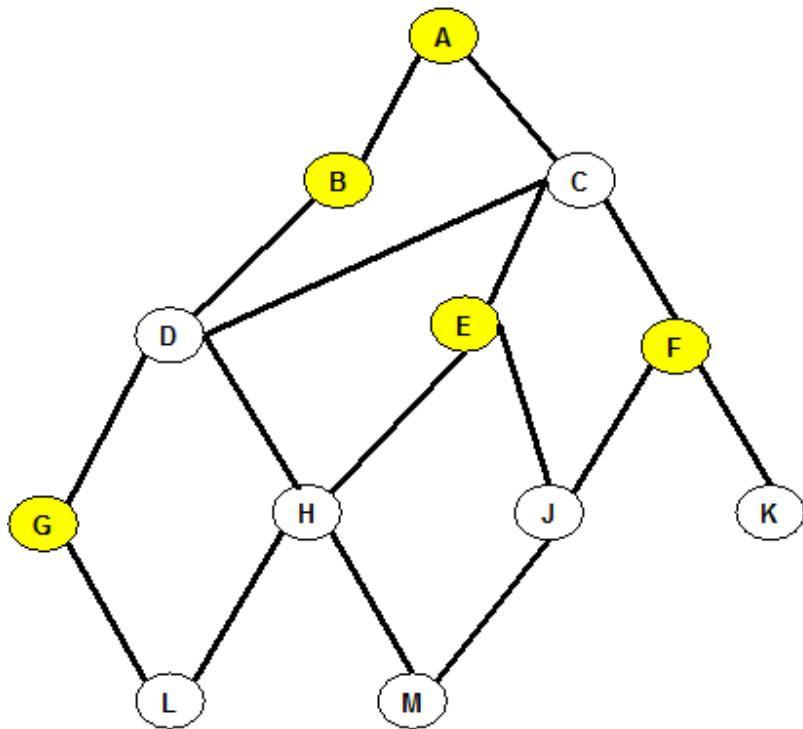


Figure 15: Example hierarchy with comprehensive list of primitive supertypes

Table 8: Primitive Supertypes in the Example Hierarchy

Concept	Comprehensive list of <i>primitive</i> supertypes
A	A
B	A, B
C	A
D	A, B
E	A, E
F	A, F
G	A, B, G
H	A, B, E
J	A, E, F
K	A, F
L	A, B, E, G

Concept	Comprehensive list of <i>primitive supertypes</i>
M	A, B, E, F

 **Note:**

1. In this view the definitions of *primitive concepts* should implicitly or explicitly include a reference to the defined *concept* itself. This is because a *primitive concept* expresses some meaning that is not fully distinguished from its supertypes by other defining *relationships*. The reference to self need not be explicitly stored and provided that it is included implicitly at run time.
2. All *active concepts* include the *root concept* in their *transitive closure*. The reference to root need not be explicitly stored provided that it is included implicitly at run time.

4.2.1.3.2.3.1.4 Proximal primitive supertypes (short normal view)



An inferred *concept definition* may contain *relationships* to the set of proximate *primitive supertype parents* of that *concept*. *Relationships* with *fully defined supertype ancestors* are omitted as are *relationships* with *primitive ancestors* that are also supertypes of one of proximate *primitive supertypes*.

This view can be used to test if a candidate *expression* is subsumed by a predicate *concept*. If the proximal *primitive supertype* view of the *concept definition* of the predicate includes *any concept* that is not in the comprehensive *primitive* view of the candidate *expression*, then the *expression* is not subsumed by the *concept*.

The | is a | *relationships* in the *SNOMED CT 'canonical table'* represent this view.

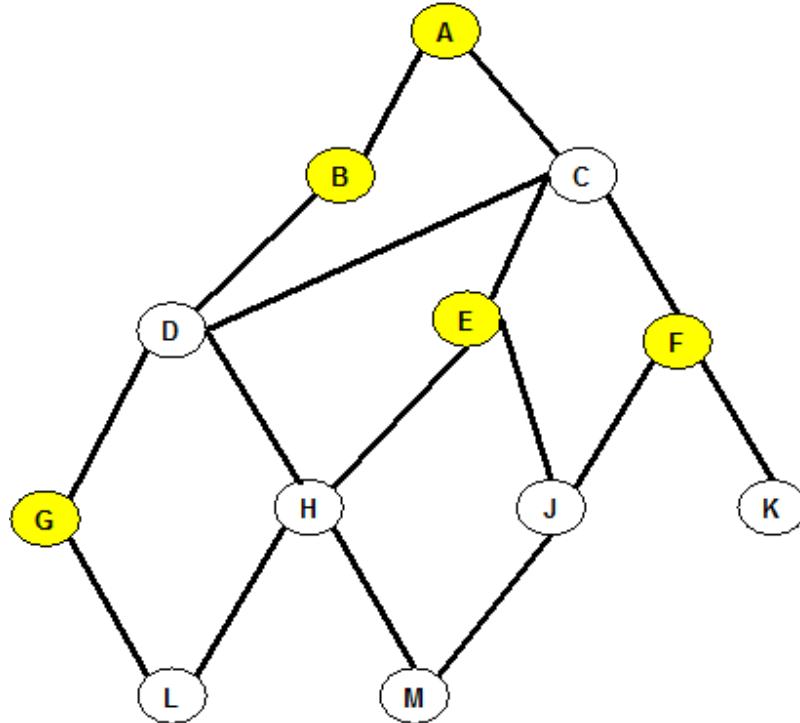


Figure 16: Example hierarchy with list of proximal primitive supertypes

Table 9: Proximal Primitive Supertypes in the Example Hierarchy

Concept	List of proximal primitives
A	A
B	B
C	A
D	B
E	E
F	F
G	G
H	B, E
J	E, F
K	F
L	E, G
M	B, E, F

👉 **Note:** The proximal primitive of a primitive concept is the concept itself.

4.2.1.3.2.3.2 Attribute aspects of concept definition views



An inferred definition view includes one of several alternative views of the *defining characteristics* of a *concept*. The considerations in this section exclude the supertype | is a | relationships.

In addition to the different views described in this section, alternative logical forms may be applied to the values of the *relationships*.

4.2.1.3.2.3.2.1 Comprehensive view of defining Relationships



An inferred *concept definition* may include all the defining *relationships* (and *relationships* groups) that are known to be true. This includes those stated and other inferred by inheritance from stated *supertype ancestors*.

The full form includes all possible *supertype ancestor* values of the stated attributes. This means that in many cases this will include a very large set of *relationships*.

Taken to its logical extreme this also includes *relationships* duplication of *relationships* with *relationship types* that are supertypes of those types stated (e.g. all | procedure site - indirect | *relationships* would be duplicated for the supertype attribute | procedure site |).

While this version of the definition model is an Abstract Logical view it is unlikely that explicit representation of this view will deliver benefits sufficient to merit this level of redundancy.

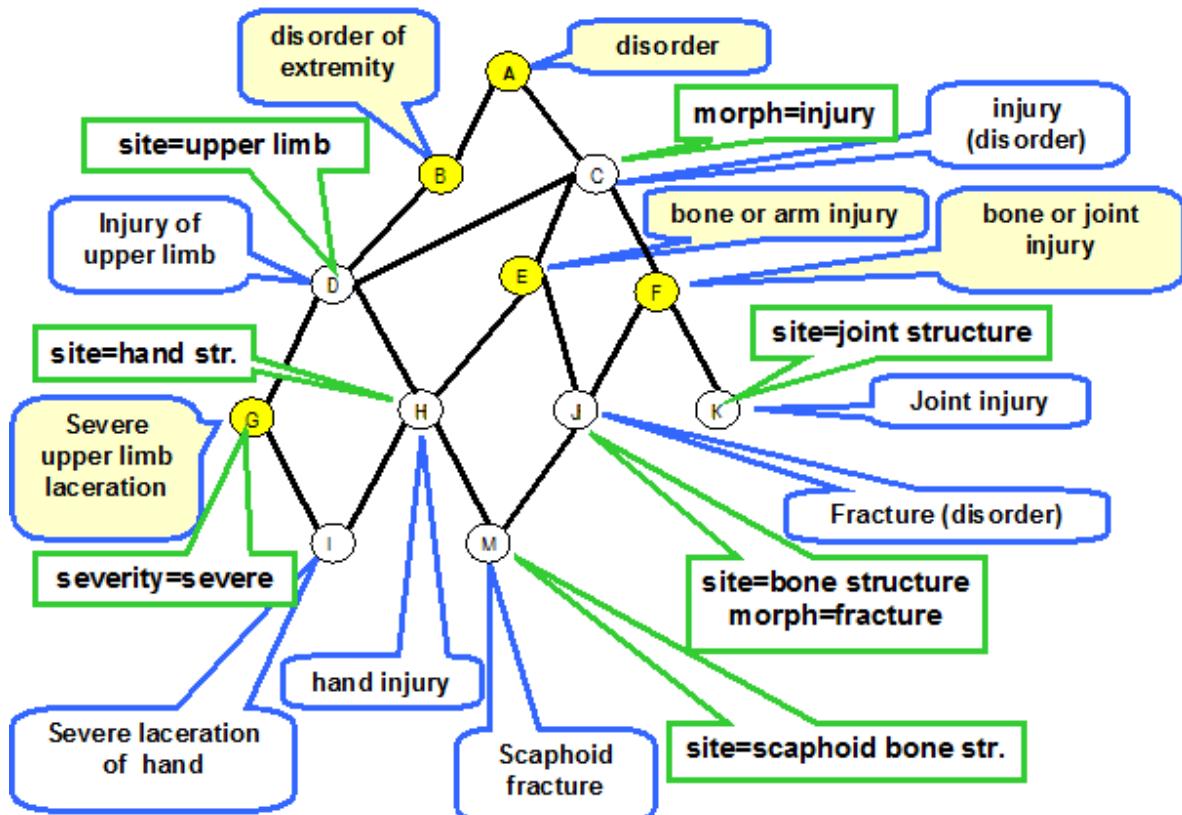


Figure 17: Illustration of sample concepts with differentiating defining characteristics

Table 10: Comprehensive attribute view of sample concepts

See [Figure 17](#)

C. Injury disorder morphology = injury	D. Injury of upper limb site = upper limb structure morphology = injury
E. Bone or arm injury (primitive) morphology = injury	F. Bone or joint injury (primitive) morphology = injury
G. Severe upper limb laceration (primitive) site =upper limb structure morphology = injury severity = severe	H. Hand injury site = upper limb structure site = hand structure morphology = injury
J. Fracture (disorder) site =bone structure morphology = injury morphology = fracture	K. Joint injury site = joint structure morphology = injury

<p>L. Severe laceration of hand </p> <p> site = upper limb structure </p> <p> site = hand structure </p> <p> morphology = injury </p> <p> severity = severe </p> <p><u>Note</u></p> <p>Although the morphology laceration is not specified in the example severe upper limb laceration refined to the site hand fully defines this <i>concept</i>.</p> <p>In a complete view (including supertypes and attributes) this difference is clear.</p>	<p>M. Scaphoid fracture </p> <p> site = upper limb structure </p> <p> site = hand structure </p> <p> site = bone structure </p> <p> site = scaphoid bone structure </p> <p> morphology = injury </p> <p> morphology = fracture </p>
--	---

4.2.1.3.2.3.2.2 Non-redundant defining Relationships ("distribution view")



An inferred *concept definition* may include the set of non-redundant defining *relationships* (and *relationships groups*) that are known to be true. This includes those stated and others inferred by inheritance from stated *supertype ancestors*. However, any *relationships* (or *relationships groups*) that are supertypes of other *relationships* (or *relationship groups*) are redundant and are not included in this view.

A *relationship* that is part of a *relationship group* is only regarded as redundant if the *relationship group* as a whole subsumes another *relationship group*.

This is the view expressed in the standard *SNOMED CT* distribution and this same view also forms part of the long *normal form*.

Table 11: Non-redundant attribute views of sample concepts

See [Figure 17](#)

<p>C. Injury disorder </p> <p> morphology = injury </p>	<p>D. Injury of upper limb </p> <p> site = upper limb structure </p> <p> morphology = injury </p>
<p>E. Bone or arm injury (primitive)</p> <p> morphology = injury </p>	<p>F. Bone or joint injury (primitive)</p> <p> morphology = injury </p>
<p>G. Severe upper limb laceration (primitive)</p> <p> site = upper limb structure </p> <p> morphology = injury </p> <p> severity = severe </p>	<p>H. Hand injury </p> <p> site = hand structure </p> <p> morphology = injury </p>
<p>J. Fracture (disorder) </p> <p> site = bone structure </p> <p> morphology = fracture </p>	<p>K. Joint injury </p> <p> site = joint structure </p> <p> morphology = injury </p>

<p>L. Severe laceration of hand </p> <p> site = hand structure </p> <p> morphology = injury </p> <p> severity = severe </p> <p><u>Note</u></p> <p>Although the morphology laceration is not specified in the example severe upper limb laceration refined to the site hand fully defines this <i>concept</i>.</p> <p>In a complete view (including supertypes and attributes) this difference is clear.</p>	<p>M. Scaphoid fracture </p> <p> site = scaphoid bone structure </p> <p> morphology = fracture </p>
--	---

4.2.1.3.2.3.2.3 Primitive differential attribute view of concept definitions



The *primitive* differential view includes only non-redundant defining *relationships* (and *relationship groups*) that are not present in the sum of the definitions of the set of *primitive* supertype *concepts*. This view provides a minimal attribute view which is semantically complete when combined with one of the *primitive* supertype views.

A *relationship* that is part of a *relationship group* is only regarded as redundant if the *relationship group* as a whole subsumes another *relationship group*.

Table 12: Primitive differential attribute views of sample concepts

See [Figure 17](#)

<p>C. Injury disorder </p> <p> morphology = injury </p>	<p>D. Injury of upper limb </p> <p> site = upper limb structure </p> <p> morphology = injury </p>
<p>E. Bone or arm injury (<i>primitive</i>)</p> <p>(morphology = injury) note 2</p>	<p>F. Bone or joint injury (<i>primitive</i>)</p> <p>(morphology = injury) note 2</p>
<p>G. Severe upper limb laceration (<i>primitive</i>)</p> <p>(site = upper limb structure </p> <p> morphology = injury </p> <p> severity =severe) note 2</p>	<p>H. Hand injury </p> <p> site = hand structure </p> <p> morphology = injury </p>
<p>J. Fracture (disorder) </p> <p> site = bone structure </p> <p> morphology = fracture </p>	<p>K. Joint injury </p> <p> site = joint structure </p> <p> morphology = injury </p>
<p>L. Severe laceration of hand </p> <p> site = hand structure </p>	<p>M. Scaphoid fracture </p> <p> site = scaphoid bone structure </p> <p> morphology = fracture </p>

 **Note:**

1. This is the attribute view expressed in the *SNOMED CT canonical form table*.
2. If the *primitive supertype* view of *primitive concepts* includes the *concept itself* (i.e. as its own proximal *primitive*) then the differential attribute view is empty for all *primitive concepts*. The entries shown above for *primitive concept* apply only where the *concept itself* is excluded from the proximal *primitive supertype* view.

4.2.1.3.2.3.2.4 Supertype differential attribute view of concept definitions



The supertype differential view includes only non-redundant defining *relationships* (and *relationship groups*) that are not present in the sum of the definitions of the supertypes of the *concept*. This view provides a minimal attribute view which is semantically complete when combined with the proximal or complete supertype view.

A *relationship* that is part of a *relationship group* is only regarded as redundant if the *relationship group* as a whole subsumes another *relationship group*.

Table 13: Supertype differential attribute views of sample concepts

See [Figure 17](#)

C. Injury disorder morphology = injury	D. Injury of upper limb site = upper limb structure morphology = injury
E. Bone or arm injury (<i>primitive</i>)	F. Bone or joint injury (<i>primitive</i>)
G. Severe upper limb laceration (<i>primitive</i>) severity = severe	H. Hand injury site = hand structure
J. Fracture (disorder) site = bone structure morphology = fracture	K. Joint injury site = joint structure
L. Severe laceration of hand <i>None</i> <u>Note</u> All distinguishing characteristics are inherited from one or both of the supertypes.	M. Scaphoid fracture site = scaphoid bone structure

4.2.1.3.2.3.3 The Short Canonical Form



The short *canonical form* is an alternative view of the *Relationships* that is provided as an RF1 release file. It consists of the union of the following two views:

- [Proximal primitive supertypes \(short normal view\)](#)
- [Primitive differential attribute view of concept definitions](#).

4.2.1.3.3 Nature of the definition



A *concept definition* has one of the following two forms:

1. fully defined concepts :

- The definition is complete. It contains *relationships* that represent the full set of *necessary* and *sufficient* conditions.

2. primitive concepts :

- The definition is incomplete. It contains *relationships* that represent a set of *necessary* conditions but this set of conditions is not *sufficient* to fully define the *concept*.

 **Note:** A *necessary* condition is a characteristic that is always true of a *concept*.

 **Example:** | morphology | = | fracture | is a necessary condition of | fracture of femur |.

 **Note:** If all members of a *sufficient* set of conditions are true they imply that the *concept* is also true.

 **Example:** | morphology | = | fracture | and | finding site | = | bone structure of femur | form a *sufficient* set of conditions that define the *concept* | fracture of femur |.

 **Note:** All members of the set of sufficient conditions are also necessary conditions. However, some necessary conditions may not form part of the *sufficient* set of conditions.

 **Example:**

Consider the *concept* | gastric ulcer |

- The | finding site | = | gastric mucosa | is a *necessary* condition for | gastric ulcer |:
 - This is true because all gastric ulcers necessarily involve the | gastric mucosa |
- The definition | morphology | = | ulcer | and | finding site | = | stomach structure | is a *sufficient* definition for | gastric ulcer |:
 - This is true because any ulcer in a stomach structure is a | gastric ulcer |.
- Therefore, an assertion that a person has an | ulcer | with | finding site | | stomach | is *sufficient* to imply that they have a | gastric ulcer |:
 - Since a gastric ulcer *necessarily* involves the | gastric mucosa | it should be possible to deduce that a person with an "ulcer" with finding site | stomach | has a disorder of with a site | gastric mucosa |.

4.2.1.3.3.1 Sufficient definition



A *sufficient* definition consists of a set of defining *relationships* (and *relationship groups*) which taken together imply a particular meaning.

The value of a *sufficient* definition is that it allows post coordinated *expression* that is sufficient to define a *concept* to be recognized as equivalent to (or a *subtype* of) a defined *concept*.

For example:

Gastric ulcer is defined as follows and this is a *sufficient* definition because any | ulcer | in a | stomach structure | is by definition a | gastric ulcer |.

116680003 | is a | =64572001 | disease | {116676008 | associated morphology | =56208002 | ulcer | ,363698007 | finding site | =69695003 | stomach structure |}

Based on this definition:

Any *postcoordinated expression* that specified a disease involving an | ulcer | with | finding site | | stomach | would be equivalent to or a *subtype* of | gastric ulcer |.

However, a *query* for all disorders involving | gastric mucosa | would incorrectly exclude the *concept* | gastric ulcer | as the site is not specified as some stomach structure rather than specifically identifying the gastric mucosa.

4.2.1.3.3.2 Necessary definition



A *necessary definition* consists of a set of defining *relationships* (and *relationship groups*) which express all the attributes that are necessarily true about a *concept* for a given version of the *SNOMED CT Concept Model*.

A *necessary definition* may contain *relationships* or *refinements* that are not essential for a *sufficient definition*.

The value of a *necessary definition* is that it allows more refined subsumption queries to be appropriately evaluated.

For example:

Gastric ulcer could be defined as follows:

```
116680003 | is a | =64572001 | disease | {116676008 | associated morphology | =56208002 | ulcer | ,363698007 | finding site |78653002 | gastric mucous membrane structure |}
```

This more tightly defined definition contains a *necessary definition* (| finding site | = | gastric mucous membrane structure |). This is necessarily true if the sufficient definition (| finding site | = | stomach structure |) is true, because any ulcer in a stomach structure is by definition a gastric ulcer.

4.2.1.3.3.3 Limitations of the current SNOMED CT model



The *current SNOMED CT model* and distribution format do not distinguish between *relationships* that are *necessary conditions* and those that are part of a set of *necessary and sufficient conditions*. For any *fully defined concepts* the set of defining *relationships* are regarded as *necessary and sufficient*.

As a result of this limitation some currently released *fully defined concept definitions* may include conditions that are *necessarily true* but are not required as part of the set of *sufficient conditions*.

👉 **Example:** Consider the two definitions shown below:

```
116680003 | is a | =64572001 | disease | , 246075003 | Causative agent |=113858008 | mycobacterium tuberculosis complex | { 116676008 | associated morphology |=6266001 | granulomatous inflammation | , 363698007 | finding site |=39352004 | joint structure |}
```

Figure 18: | tuberculous arthritis |

```
116680003 | is a | =64572001 | disease | , 246075003 | causative agent |=41146007 | bacteria | { 116676008 | associated morphology |=23583003 | inflammation |, 363698007 | finding site |=39352004 | joint structure |}
```

Figure 19: | bacterial arthritis |

The definition of | tuberculous arthritis | differs from that of | bacterial arthritis | in two respects. In practice the first of these (| causative agent |= | mycobacterium tuberculosis complex |) is sufficient to define the *concept*. However, the nature of the inflammation that results is, necessarily, granulomatous.

Thus an *expression* that specifies | bacterial arthritis | with | causative agent |= | mycobacterium tuberculosis complex | is clinically equivalent to the *concept* | tuberculous arthritis | even though it does not explicitly refine the nature of the inflammation.

In contrast the *current SNOMED CT model* computes | bacterial arthritis | with | causative agent |= | mycobacterium tuberculosis complex | as supertype of | tuberculous arthritis |. This occurs because the *expression* | bacterial arthritis | with | causative agent |= | mycobacterium tuberculosis complex | does not specify of the nature of the inflammation.

👉 **Future enhancements:** Options for distinguishing the sufficient set of defining *relationships* from those that are merely necessarily true are being investigated. A complete solution to this issue needs to support

the recognition of several separate sufficient sets. However, initially a solution recognizing a single sufficient set may be introduced.

4.2.1.3.3.4 Impact on retrieval



A *necessary* definition is inevitably more complete than a *sufficient* definition. From the perspective of retrieval the completeness of a definition is a mixed blessing.

- It is an advantage for candidate *expressions* as they will be subsumed by a wider set of appropriate predicates.
- It is a disadvantage for a predicate *expression*, the necessary conditions may result in incomplete retrieval. A candidate *expression* that satisfies all the *sufficient* conditions should be included. However, it will be excluded unless it satisfies all the necessary conditions in the predicate.

This occurs where the definition of a *concept* states conditions that are *necessarily* true but which go beyond those that are *sufficient* to distinguish a *concept* from its supertypes.

Example:

The *normal form* definition of | pulmonary tuberculosis | is as follows:

```
116680003 | is a | = 64572001 | disease |
,246075003 | causative agent | = 113858008 | mycobacterium tuberculosis complex |
{116676008 | associated morphology | = 6266001 | granulomatous inflammation |
,363698007 | finding site | = 39607008 | lung structure | }
```

Used as a *query predicate*, this will exclude valid candidate *expressions* such as ...

```
233604007 | pneumonia | : 246075003 | causative agent | = 113861009 | mycobacterium tuberculosis |
```

- This *expression* is not subsumed by the full definition of | pulmonary tuberculosis | because it does not mention "granulomatous inflammation". This type of inflammation is characteristic of "mycobacterium tuberculosis" infection and so is necessarily present. Since currently *SNOMED CT* definitions do not distinguish the sufficient and necessary conditions this cannot be inferred.

A more inclusive *query predicate* that specifies a sufficient set of conditions for | pulmonary tuberculosis | can be constructed by removing the morphology condition.

```
116680003 | is a | = 64572001 | disease |
,246075003 | causative agent | = 113858008 | mycobacterium tuberculosis complex |
,363698007 | finding site | = 39607008 | lung structure |
```

- This correctly subsumes both the *precoordinated concept* | pulmonary tuberculosis | and the *postcoordinated* candidate *expression* above.

Note: To ensure complete retrieval

- When selecting a *concept* as part of a *query predicate*, view its *normal form* definition and decide whether some of the conditions should be omitted;
- Specify the minimum set of conditions sufficient for the intended purpose.

Future enhancements: In future, when the *SNOMED CT* model is revised to distinguish *sufficient* sets of defining *Relationships*, the sufficient definition can be used as the predicate for a retrieval. A candidate *expression* matches a predicate if it *necessarily* fulfills all the *sufficient* conditions specified in the *query*.

4.2.2 Logical Model of SNOMED CT expressions



Figure 20 shows the general abstract model for a *SNOMED CT expression*. This diagram also shows the references between *expressions* and *components*.

An *expression* is a collection of references to one or more *concepts*. The *expression* consists one or more *focus concepts* and an optional *refinement*.

The *focus concept* and the names of the refining attributes are represented by references to *SNOMED CT concepts*. The value of a refining attribute is itself an *expression* and is structured in the same way. Thus nested *expression* can be used to refine the value of a refining attribute.

An *expression* represents an instance of the meaning defined by the defining *relationships* of the *focus concepts* as modified by the *refinements*.

The meaning of each *refinement* is expressed by an *attribute name* which names a property and an *attribute-value* which expresses the value of that property.

- The *attribute name* must be a *concept* that is a *subtype* of $|attribute|$.
- The *refinement* value may be a *concept* or *expression* that is appropriate to the named attribute. The values that are appropriate to an attribute are specified by the *Concept Model*. In most cases, any *subtype* of a *concept* that is permitted as a value of an attribute is also permitted.
- *Refinements* may be grouped to represent interdependencies between them in the same way as *relationship groups*.

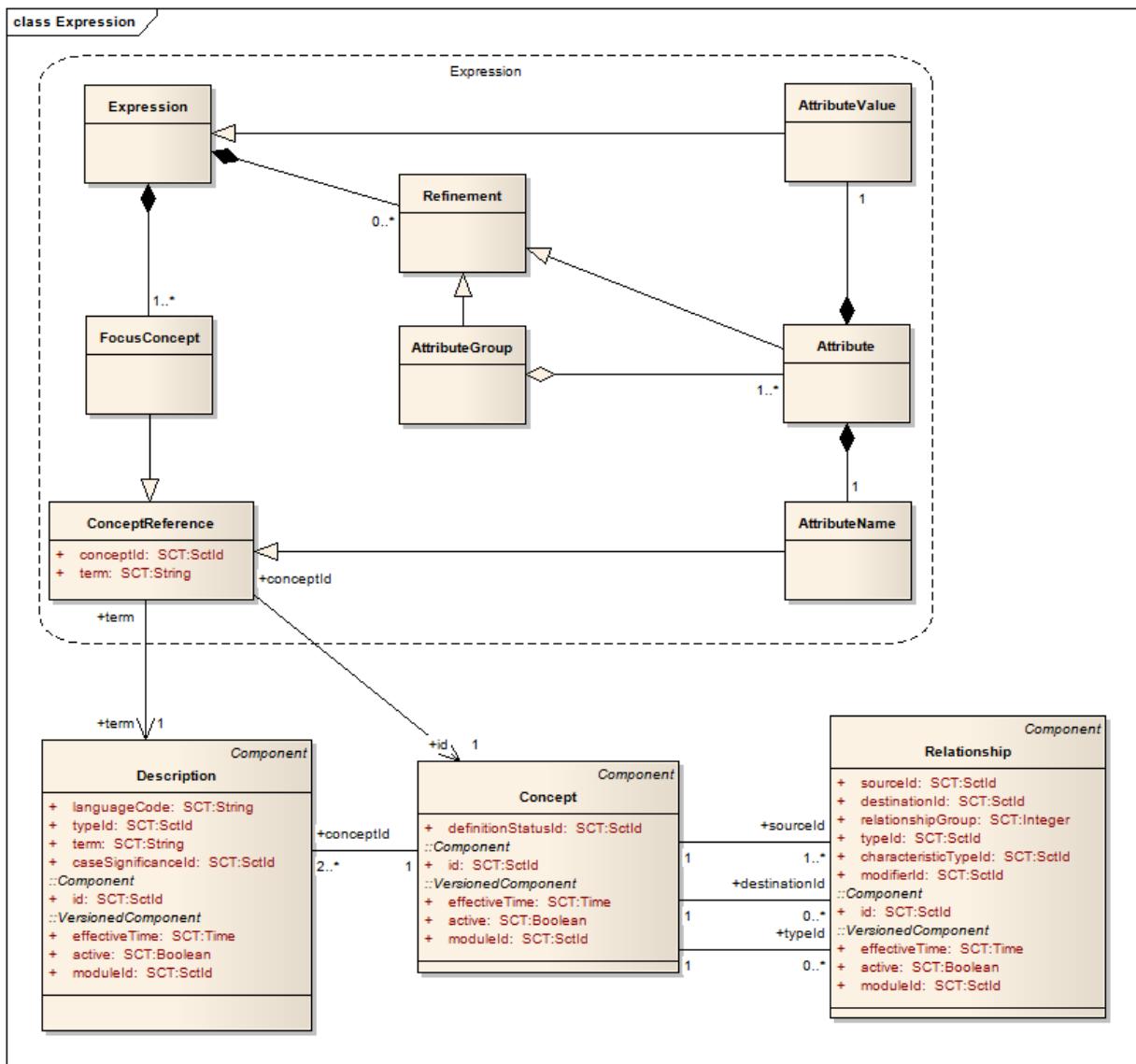


Figure 20: General Abstract Logical Model of a SNOMED CT expression

4.2.2.1 Refinement



An *expression* represents an instance of the meaning defined of the *focus concepts* are as modified by the *refinements*.

Various types of *refinement* are possible. Of these some are fully supported by the *SNOMED CT Concept Model* and released data while other possible methods of *refinement* step outside those boundaries.

4.2.2.1.1 Refinement of defining Relationships

4.2.2.1.1.1 Refinement individual attribute values



A defining *relationship* of the base *concept* can be refined by applying a value that is a *subtype* of the defining value.

This approach to *refinement* is fully supported by the *SNOMED CT Concept Model*.

Defining *relationships* that are marked with the *refinability* property value "not refinable" should not be refined.

4.2.2.1.1.2 Refinement attribute names



A defining *relationship* of the base *concept* can also be refined by applying a name that is a *subtype* of the defining *attribute name*. For example, if the defining *relationship* specifies a | procedure site | this may be refined to | procedure site - direct | or | procedure site - indirect |.

4.2.2.1.1.3 Refinement of defining Relationship groups



If a *refinement* is applied to one of the defining *relationships* within a *relationship group*, it is the group a whole that is refined.

It is also permissible for a stated (close-to-user) *expression* to refine a *relationship* without grouping the refined *relationship* or without fully enumerating the group of which it is part. In this case, resolution to an inferred structure should apply the ungrouped *relationship* value (or partially enumerated group) as a *refinement* of any group to which that *refinement* can be appropriately applied.

4.2.2.1.1.4 Nested refinement of defining Relationships



The value of a defining *relationship* may itself be refined. In this case the value of the *relationship* becomes a *postcoordinated expression* rather than a *precoordinated concept*.

This occurs most frequently in the following situations:

Laterality refinement

The laterality qualification applies to the value of the | procedure site | or | finding site | *relationship* and is logically nested under site.

(Note lateralization is discussed separately)

Refinement of situation with explicit contexts

The | associated finding | or | associated procedure | is a | clinical finding | or | procedure |, which may itself be refined (e.g. with severity).

4.2.2.1.2 Applying values to qualifiers

4.2.2.1.2.1 Applying values to individual qualifiers



A qualifying *relationship* of the base *concept* can be used to apply a *refinement*.

The nature of the allowable *refinement* using *qualifiers* is determined by the value of the "refinability" property of the qualifying *relationship*.

Not refinable

The *qualifier* can only be used to refine the base *concept* by applying the qualifying value specified in the distributed table.

Refinable

The *qualifier* can be used to refine the base *concept* by applying the qualifying value specified in the distributed table or any *subtype* of that value.

Mandatory to refine

The *qualifier* can be used to refine the base *concept* by applying a *subtype* of the qualifying value specified in the distributed table. It cannot be applied with the specified value itself as this is a non-specific grouping value for possible *refinements*.

This approach to *refinement* is fully supported by the *SNOMED CT Concept Model*.

4.2.2.1.2.2 Grouping qualifier refinements



In theory the value of a *qualifier* may apply only to the content of one *relationship group*.

Currently *qualifiers* are not grouped in *SNOMED CT releases* and therefore grouping of *qualifier refinements* is not supported in the *current Concept Model*. However, this is under review and the model may be extended to include grouped *qualifiers* in future. This review is required because problems arise with subsumption

testing where *precoordinated* definitions include grouped *attribute-value pairs* and the *expression* uses an ungrouped, *qualifier* derived, attribute.

4.2.2.1.2.3 Nested refinement of qualifiers



The value of a *qualifier* may itself be refined and represented as an *expression* rather than a *precoordinated concept*.

This occurs most frequently with *expressions* which qualify high level "*situation with explicit context*" *concepts* (e.g. "finding with *explicit context*"). In this case the | associated finding | is applied as a *qualifier* which may itself be refined (e.g. with severity).

4.2.2.1.3 Applying laterality to a concept



A laterality value (left, right or bilateral) can be applied as a *qualifier* to lateralizable body structure *concepts*.

It is also permissible for a stated (close-to-user) *expression* to lateralize a base *concept* that has a definition including reference to a lateralizable body structure. In this case, resolution to an inferred structure should apply the laterality to all values in the base *concept definition* that are lateralizable body structures.

This approach is fully supported by the *SNOMED CT Concept Model*, provided that appropriate *transforms* are applied.

Note

If lateralization is specific to particular aspects of the *concept* then the laterality should be applied to the appropriate *relationship* as part of a nested *expression*.

4.2.2.1.4 Sanctioned and unsanctioned refinement

4.2.2.1.4.1 Introduction to refinement sanctioning



SNOMED CT relationships provide information that may be used to determine the types of *refinements* can be processed to determine *equivalence* and *subsumption*. However, even where a *concept* has no specific *relationship* it is possible to apply a refinement using an *attribute* that the *Concept Model* permits for *concepts* in that domain. Other *attributes* are not recommended for refinement as they will result in *expression* that cannot be normalized or reliably compared. Specific issues with unsanctioned refinements are considered in:

- [Unsanctioned use of "Concept Model attributes"](#);
- [Use of "unapproved attributes"](#);
- [Advantages and disadvantages of unsanctioned refinements](#).

4.2.2.1.4.2 Unsanctioned use of "Concept Model attributes"



In some situations it may seem to be useful to use one of the attributes used in the *SNOMED CT Concept Model* to refine a *concept* that does not have a defining *relationship* or *qualifier* named by this attribute.

Provided that this is limited to qualifications that the *Concept Model* specifies for *concepts* of the same general type this approach can be applied. However, *Concept Model* attributes should not be applied to *concepts* of other types (for example the "approach" attribute should not be applied to a "finding"). Use of unsanctioned (but 'allowable') attributes for *refinement* may limit semantic interoperability.

Despite this limitation it may be appropriate to use a community agreed approach for a particular defined purposes. However, care should be taken to use attributes only in the manner described in the [Concept Model Guide \(6\)](#).

4.2.2.1.4.3 Use of "unapproved attributes"



The *SNOMED CT release* also includes a large number of attributes that are classified as "unapproved attributes".

Most of these originate from earlier terminology efforts. They have as yet not been applied in the *SNOMED CT Concept Model* and there is no guarantee that they will be used in a particular manner in the future.

This approach is not supported by the *SNOMED CT Concept Model*. Therefore any use of unapproved attributes for *refinement* is likely to limit semantic interoperability.

Despite these limitations, it may be appropriate to use a community agreed *Reference Set* of unapproved attributes within a defined user community for a particular defined purpose. Any such use should be fully documented by those responsible for its adoption. In the future as the *SNOMED CT Concept Model* evolves, additional supported attributes may provide a *migration* path for information recorded using a well-documented set of rules for a limited set of use cases.

4.2.2.1.4.4 Advantages and disadvantages of unsanctioned refinements



Note: THIS SECTION CONTAINS DISCUSSION NOTES ONLY.

The presence of defining or qualifying *relationships* certainly simplifies the task of implementing facilities for *refinement*. It also provides an indication that subsumption and *equivalence* computation may be possible. However, at this stage there is no definitive view of the extent to which *SNOMED CT* should sanction and permit particular *refinements* while deprecating or prohibiting other *refinements*.

Disadvantages of prohibition of all unsanctioned refinements

- **Lack of ability to express some required meanings:**
 - Until an attribute is included in the *Concept Model* and appropriately populated for all relevant *concepts*, it cannot be used to refine some *concepts* that might reasonably be so refined. The consequence of this are an inability to express some meanings required by users with approved *SNOMED CT* *expressions*.
 - One example of this is that at present the following *expression* would not be sanctioned as headache has no associated severity *qualifier*. While this looks like an error that could readily be corrected it serves to illustrate the point.

25064002 | headache |:246112005 | severity |=24484000 | severe |

Disadvantage of allowing unconstrained refinement

- **Nonsense expressions with no "sensible" meaning:**
 - e.g. 25064002 | headache |:103366001 | with color |=414497003 | infra-red |
 - These are probably not a major cause for concern because it is impossible to create a foolproof approach that guarantees that all *expressions* will be sensible:
 - The following nonsense example is "sanctioned" in the sense that the site specified is a *refinement* of | head structure | which is the defined finding site for | headache |:
 - 25064002 | headache |: 363698007 | finding site |= 87056002 | infantile diploetic mastoid cell |
 - A nonsense *expression* is meaningless and where it is subsumed is largely irrelevant. Ideally it would subsume under nonsense *expressions* but that would require a knowledge of the rationality of all possible *expressions*.
 - In the absence of a tractable way to prohibit nonsense, avoidance and management of nonsense is an issue for implementers, users and qualify reviewers.
- **Nonsense expressions which may express a superficial "sensible" meaning:**
 - e.g. 25064002 | headache |:103366001 | with color |=301888000 | pale color |
 - A person reading this might think this expresses the fact the person's head (or face) was pale at the time of the headache. Logically in *SNOMED CT* it would mean that the headache is pale in color which is nonsense. However, an argument could be advanced that the same rules apply as those for indirect laterality and thus this could *transform* to:
 - 25064002 | aching pain |: 363698007 | finding site |= (69536005 | head structure |:103366001 | with color |=301888000 | pale color |).

- This is still nonsense from a *SNOMED CT* perspective or perhaps it could correctly mean is | aching pain in the pale colored head structure |. However, if the author (or authoring application) assigned such an *expression* to represent two distinct findings | headache | and "head is pale in color" this meaning would not be apparent from a logical computational perspective.
- While prohibition of nonsense is not tractable it may be feasible to state rules that express which forms of *expression* are logical and computable. Furthermore the outcome of these rules needs to be deterministic so that the result of transforming do not differ according to implementation.
- **Alternative rational *expressions* of similar meanings :**
 - Consider the following:
 1. 25064002 | headache |: 279114001 | character of pain |=410704005 | throbbing sensation quality |
 2. 162308004 | throbbing headache |
 3. code=162306000 | headache character | value =410704005 | throbbing sensation quality |
 - This assumes an information model with an observable entity *concept* naming a value in a separate information model attribute (*HL7 Observation* supports this).
 4. 29695002 | throbbing pain |:363698007 | finding site |=69536005 | head structure |
 5. 25064002 | headache |:162306000 | headache character |=410704005 | throbbing sensation quality |.
 - All these *expressions* appear rational but only options 2 and 4 have the same *normal form* in the present *SNOMED CT Concept Model*.
 - Potentially option 3 could also be computed if both (a) the information model *terminology model*/interface was clear and (b) the *SNOMED CT* definition of 162308004 | throbbing headache | is enhanced to add "363713009|has interpretation|=410704005|throbbing sensation quality".
 - On the other hand option 1 is more in line with the way disorders are refined by "severity" and other qualitative *refinements*. For this to be computable equivalent the *concepts* "29695002|throbbing pain|" and 162308004 | throbbing headache | would both need revised definitions in which they were defined as having "279114001|character of pain|=410704005|throbbing sensation quality|".
 - Option 5 also looks superficially reasonable and shares the general feel of option 3. However, since 162306000 | headache character | is an "observable entity" rather than an "attribute" this representation would be contrary to one fundamental principle of *refinement* - that the name of the *refinement* should be a *subtype* of the *concept*"attribute". This means *current normal transform* rules would not result in a proper *normal form* and indeed might reasonable report an error.
- **User interface design issues :**
 - Given all of the above points, application designers will struggle to create sensible and consistent interfaces unless advice on sanctioning is provided.
 - Different issues will apply according to the nature of the interface. For example this may include:
 - What options to offer users to allow *refinement* of specific *concepts*;
 - How to represent the meaning that results from selecting options on a structured data entry form as a *SNOMED CT expression*;
 - How to encode meaning derived from *natural language processing*.

Interim recommendations

1. Wherever refining an existing defining or qualifying *relationship* enables representation of the required meaning this approach should be preferred.
2. Where 1 does not meet the requirements any attribute which is used in the *concept model* for *concepts* of the same type may be applied. The value applied to the attribute must be one of the allowable values as specified for that attribute in the [Concept Model Guide \(6\)](#):

- For example a | causative agent | attribute can be applied to a clinical finding *concept*. The value assigned to this attribute is a value assigned from | Organism |, "physical force", "physical object" or | substance |. However, | causative agent | cannot be applied to refine a procedure. Furthermore the value of the | causative agent | cannot be a procedure or disorder.

3. Where neither 1 nor 2 meet the requirement use of additional attributes or values may be considered to meet a specific requirement. However, in this case, the implementer and/or user community will need to:

- Avoid a direct conflict with other uses of the same attribute.
- Ambiguity will arise if an existing attribute is overloaded to fulfill a different use-case:

 **Example:** The | laterality | attribute is used in the *concept model* to specify which of two functionally symmetrical paired structures is involved (e.g. "left wrist", "right kidney"). It should not be used for:

- non-symmetrical structures (e.g. heart structures where the use of | left | and | right | refers to functionally different structures).
- right or left side of a midline structure (e.g. | head | :| laterality | = | left | does not mean the "left side of the head" it means "left head" - and is thus not a useful *refinement*).
- relative laterality (e.g. | trachea | :| laterality | = | left | does not mean "to the left of the trachea" or "trachea deviated to the left" it means "left trachea" - and is not a useful *refinement*).

- Agree the approach to be taken in advance:
 - Ad-hoc *refinement* by end-users without any guidance on an agreed approach is liable to lead to multiple ways of representing the same required meaning and a loss of interoperability.
- Document the approach taken in forms that:
 - Allow consistent use within the community;
 - Identify any issues related to computation of *equivalence* and subsumption between these local variant *expressions* and the content of *SNOMED CT*;
 - Are communicated to an appropriate *SNOMED CT* Working Group to help establish a wider consensus.
- Make provision for future *migration* of data as a common *SNOMED CT* approach is developed in future.

 **Note:** Within the UK, NHS Connecting for Health has issued guidance on post coordination which specifies *constraints* on allowable *refinements* and adds some specific *extensions* to the *refinements* sanctioned by released *relationships*. These guidance documents are available to implementers in the UK.

4.2.2.1.5 Applying values to concepts



Information model attributes such as values applied to an observable, also effectively refine the meaning of the *concept* as used in the record.

Currently the *SNOMED CT Concept Model* does not address issues of *equivalence* between a particular value applied to an observable or measurement procedure and a potentially similar finding (e.g. | creatinine measurement, serum | with a specified value and a finding such as | serum creatinine raised |).

There is a loose approximation using the | interprets | and | has interpretation | *Relationships* between some | clinical finding | *concepts* and relevant | observable entity | or | laboratory procedure | *concepts*.

 **Example:** | serum creatinine raised | has a definition that includes:

- | interprets |=| creatinine measurement, serum |
- | has interpretation |=| above reference range |

👉 **Future enhancements:** The *relationships* between | observable entity |, | laboratory procedure |, | evaluation procedure | and | clinical finding | *concepts* are currently under review.

4.2.2.2 Modeling semantic context



When a clinical finding is mentioned in a patient record certain assumptions are usually made about what it means in relation to the person who is the subject of that record. Thus if the finding | wheezing | is present in a record it is assumed to mean that the subject of that record is wheezing at the time of examination. This assumed meaning might be stated in full "the subject of the record is currently wheezing" but a contracted form that omits explicit reference to the subject, timing and presence of the finding is more usual in written records.

Similarly when a procedure is mentioned in a patient record assumptions are usually made about what it means in relation to the subject of that record. Thus, in the absence of other information, the mention of the procedure | cholecystectomy | may be assumed to mean that a "the subject of the record had a cholecystectomy at a stated time".

Although default assumptions such as those above may be made, it is also possible for mention of the same finding or procedure to have a very different meaning. For example, "past medical history of wheezing", "not wheezing", "father suffers from wheezing", | cholecystectomy planned |, | cholecystectomy not done |.

The *SNOMED CT* context model provides a way to model *concepts* that explicitly state the *clinical situation* in which they are used. This same model also allows the construction of *expressions* that explicitly state the *clinical situation* in which a *concept* is being used in a particular record.

A proprietary record structure or a *reference information model* may also express aspects of context and these can be mapped to the *SNOMED CT* context model where appropriate to create comparable *expressions*.

The context model also specifies a default context that applies to findings and procedures which are expressed in a patient record without any explicit statement of context.

The most important aspects of the context model are those which have the potential to express a meaning that differs fundamentally from the meaning associated with the default context. Changes to context that have this fundamental effect on meaning are referred to as "axis modifying". The phrase "axis modifying" indicates a change that shifts the meaning between different axes in the *subtype hierarchy*.

The context model allows "axis modification" to be expressed within the general abstract logical model applied to all *SNOMED CT concepts*. To achieve this a *concept* such as | FH: Diabetes mellitus | is modeled as a *subtype* of | family history of disorder |. It is not a *subtype* of | diabetes mellitus | but instead its association with the finding | diabetes mellitus | is modeled using a defining *relationship* | associated finding |. Similarly a | Hip replacement planned | is a *subtype* of | planned procedure | (not a *subtype* of "hip replacement"). It is related to "hip replacement" by an | associated procedure |*relationship*.

4.2.2.3 Alternative logical abstract model views of expressions



Like a *concept*, an *expression* may be logically transformed into a variety of different views taking account of the definitions of the *concepts* which it references (i.e. the *Concept Identifiers* included in the *expression*).

4.2.2.3.1 Close-to-user expression view ("stated")



The close-to-user (or "stated") view of an *expression* contains references to the *concept* (or combination of *concepts*) together with *refinements* as selected by the user or as encoded by a clinical application to represent the semantics of a single clinical statement (i.e. a discrete clinical record entry).

The close-to-user view of an *expression* is the faithful representation of the information entered. For clinical safety and accountability purposes this should be regarded as the primary stored and communicated view of clinical information encoded using *SNOMED CT*.

👉 **Note:** This view includes *refinements* applied by an application based on selections made in an entry form as well as those made explicitly. It does not include any *relationships* that are added based on *classifier* rules to make the *expression* complete or to normalize it.

4.2.2.3.2 Inferred expression views



An inferred *expression* can be derived from a stated *expression* by applying rules that take account of the definition of the refined *concept* and the associated refined values.

Inferences are drawn based on a consistent set of logical rules applied to the *expression* taking account of the definitions of *concepts* referenced by the *expression*.

Alternative semantically identical *expressions* may be generated using different logical *transformations*. The purpose of logical *transformations* is to support accurate and complete information retrieval through subsumption testing.

In general *terms* the types of *transformation* and resulting inferred views for *expression* are similar to those for *concept definitions*. The following sections of the guide identify some of inferred *expression* views and some of the differences between *expressions* and *concept definitions*.

4.2.2.3.3 Simple, nested and grouped expressions



A typical close-to-user *expression* consist of a single *concept* modified by optional *refinements* as shown in [Figure 21](#). This may look like a *concept definition* but it is not defining the *concept* | hand pain |, it is specifying a more specific meaning by refining the | finding site | of the *concept* | hand pain | and adding a severity *qualifier*.

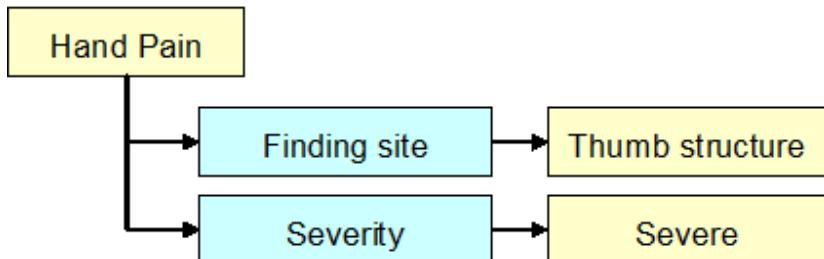


Figure 21: Refining a concept to add specificity

The target of a *refinement* may itself be refined producing a nested structure. An example of this is the application of laterality to finding site as shown in [Figure 22](#).

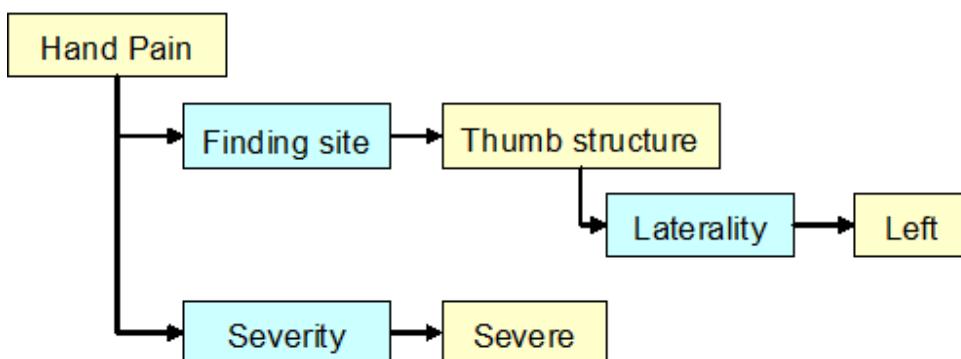


Figure 22: Nested refinement applied to a body site

In some cases, *refinements* within an *expression* may be grouped to represent association between a two different *refinements*. For example, a method and a target site or device as shown in [Figure 23](#)

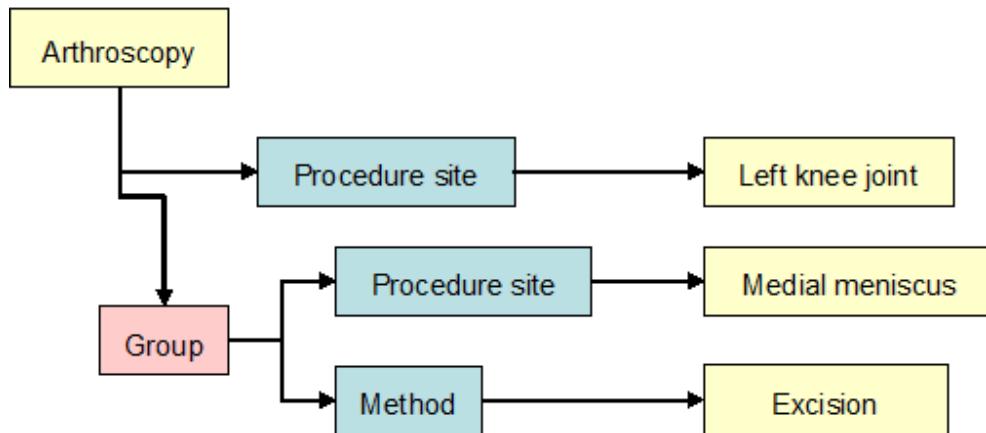


Figure 23: Grouped refinement

4.2.2.3.4 Expressions with multiple focus concepts



Some expressions may have multiple *concepts* followed by optional *refinements* as shown in [Figure 24](#)

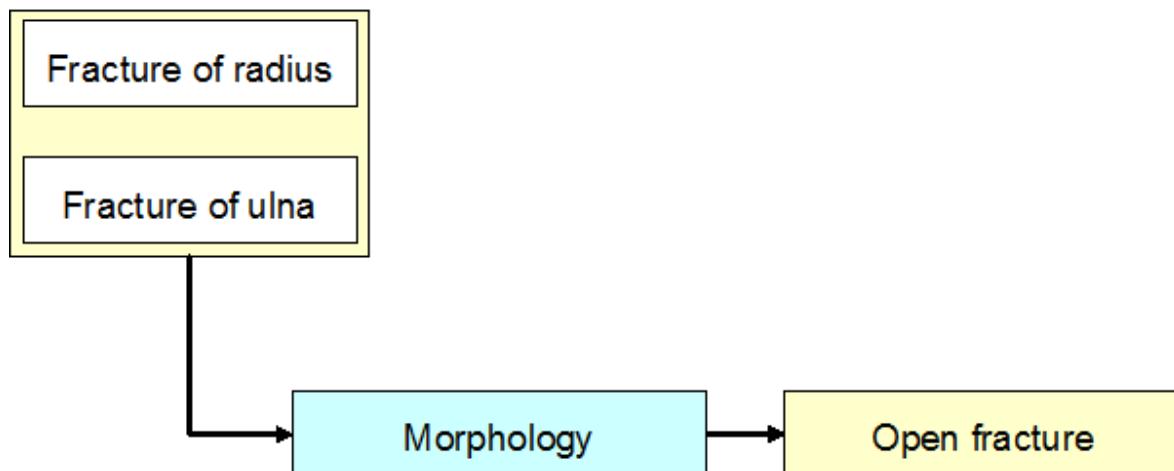


Figure 24: An expression with two focus concepts

The base of an *expression* may thus be one or more supertype *concepts* that are combined to produce a single meaning.

It is important to note that combining *concepts* at this level presumes that the result is intended to be a single combined meaning which is subsumed by the meaning of the combined *concepts*. Furthermore, the same *refinements* apply to the combined meaning of this set of *concepts*.

Some representational forms (e.g. HL7 version 3 Concept Descriptor data type) do not allow combinations to be expressed in this way. However, it is possible to apply a simple logical *transformation* to create a semantically identical view that can be conveyed in a syntax that supports a single *focus concept* with *refinements* (see [Figure 25](#)).

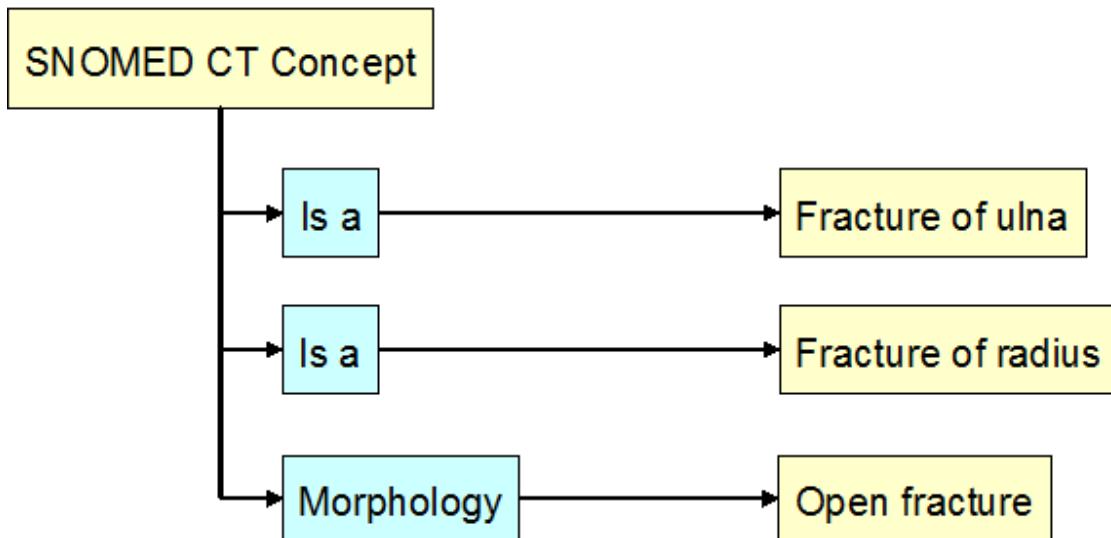


Figure 25: An alternative view of an expression with two focus concepts

4.2.2.3.5 Expressions that include context



Expressions may also explicitly represent the semantic context surrounding a finding or procedure. In these cases, the finding or procedure is nested inside the context component of the expression. The outer layer of the expression, which expresses the context, is sometimes referred to as the *context wrapper*. The nested expression representing the finding or procedure is sometimes referred to as the "clinical kernel".

Figure 26 illustrates how the general concept 281666001 | family history of disorder | can be refined to represent family history of a specific condition.

Figure 27 illustrates an alternative (computationally equivalent) representation of the same situation. In this case the family history situation is itself represented by an expression.

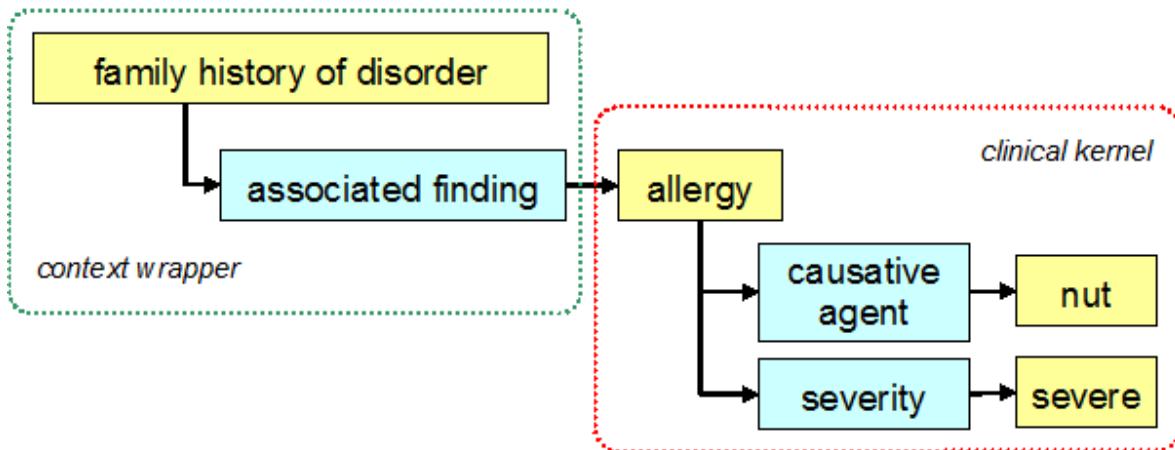


Figure 26: Family history of a specific type of severe allergy to nuts as close-to-user form expression

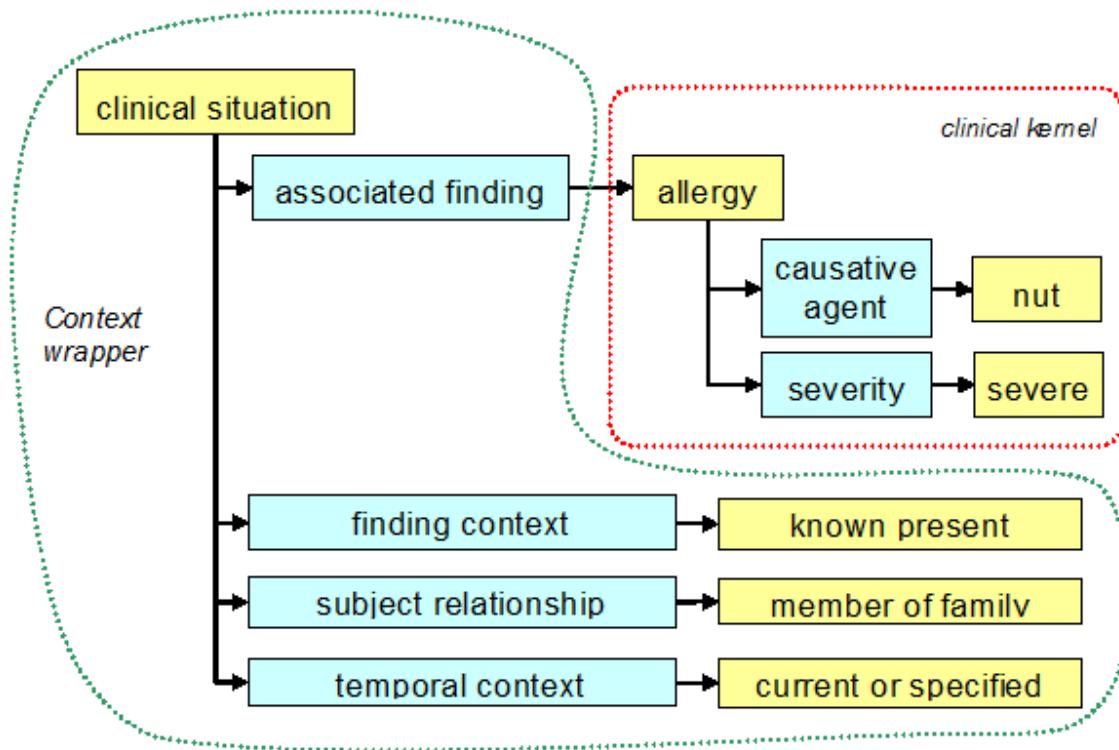


Figure 27: Family history of severe allergy to nuts represented by using a context wrapper expression

4.2.2.3.6 Normal form expression view



The theoretical range of equivalent *expressions* for a single idea includes two end-points:

- A fully *precoordinated expression* in which a single *concept identifier* is used to represent the idea;
- A maximally *postcoordinated expression* in which every facet of the idea is separately represented by an *attribute-value pair*.

In between these end points are a variable number of equivalent partially *postcoordinated expressions*.

👉 **Example:** For a detailed example see [Example of equivalent postcoordinated expressions](#) on page 84

In order to compare *expressions*, is useful to be able to transform from these varied *expression* into a common *normal form expression*. This is possible using the combination of the *expression* and the definitions of the *concepts* to which it refers. As long as a reference *concept* is *fully defined* the defining *Relationships* for that *concept* can replace the *concept identifier* in the *expression*. This process reveals redundancies that can be removed by merging the definitions with the *expression*. An end-point is reached when all the *concepts* referenced by the *expression* are *primitive*. This is referred to as the *normal form*.

The process of normalization of *expressions* is described in detail in [Transforming expressions to normal forms](#) on page 363.

👉 **Note:** The most important requirements for logical transformation of *expressions* is to enable information entered (in a close-to-user view) to be readily tested for *equivalence* or *subsumption* against another *expression* or to test inclusion within constrained range of values.

4.2.2.3.6.1 Example of equivalent postcoordinated expressions



To illustrate the range of possible equivalent expressions [Table 14](#) shows the defining characteristics of the hypothetical³ concept "red steel pedal bike" and its supertype ancestors.

Table 14: Definitions of concepts used in illustration of alternative representations

Id	Concept	Defining Characteristics
1	Device (PRIMITIVE)	is a = Thing
2	Metal device	is a = Device Made of = Metal
3	Transport device	is a = Device Used as = Transport
4	Steel transport device	is a = Transport device is a = Metal device Made of = Steel Used as = Transport
5	Pedal powered transport device	is a = Transport device Used as = Transport Used as = Transport Power = Pedals
6	Bicycle (PRIMITIVE)	is a = Transport device Used as = Transport Moves on = 2 wheels
7	Pedal bicycle	is a = Pedal powered transport device is a = Bicycle Used as = Transport Moves on = 2 wheels Power = Pedals

³ This hypothetical concept is chosen in preference to a real SNOMED CT concept to allow illustration of theoretical points with simple qualifiers. While all the points illustrated apply to some SNOMED CT concepts but there is no single concept that readily illustrates all these points without introducing other issues or having a long name that complicates the illustration.

Id	Concept	Defining Characteristics
8	Red pedal bicycle	is a =Pedal bicycle is a =Pedal powered transport device Used as = Transport Moves on = 2 wheels Power = Pedals Color = Red
9	Steel pedal bicycle	is a =Pedal bicycle is a =Steel transport device Used as = Transport Moves on = 2 wheels Power = Pedals Made of = Steel
10	Red steel pedal bike	is a = Red pedal bicycle is a = Steel pedal bicycle Used as = Transport Moves on = 2 wheels Power = Pedals Made of = Steel Color = Red

Figure 28 illustrates a range of *expressions* based on each of the *concepts* defined in might be used to represent the *concept* "red steel pedal bike".

Expression K is a *precoordinated expression* using the *concept* "10 | red steel bike| ". Each of the other forms is *postcoordinated* by adding refinements that build on the *concept* definitions shown in *Table 14*.

These *expressions* would all be equivalent if the definitions were complete and accurate. In that case, it would be possible to transform between them without losing information by appropriately adjusting the associated refinements to take account of the *concept* definitions. In practice the *concept* "bicycle" is marked as *primitive* which places a limit on transformation process.

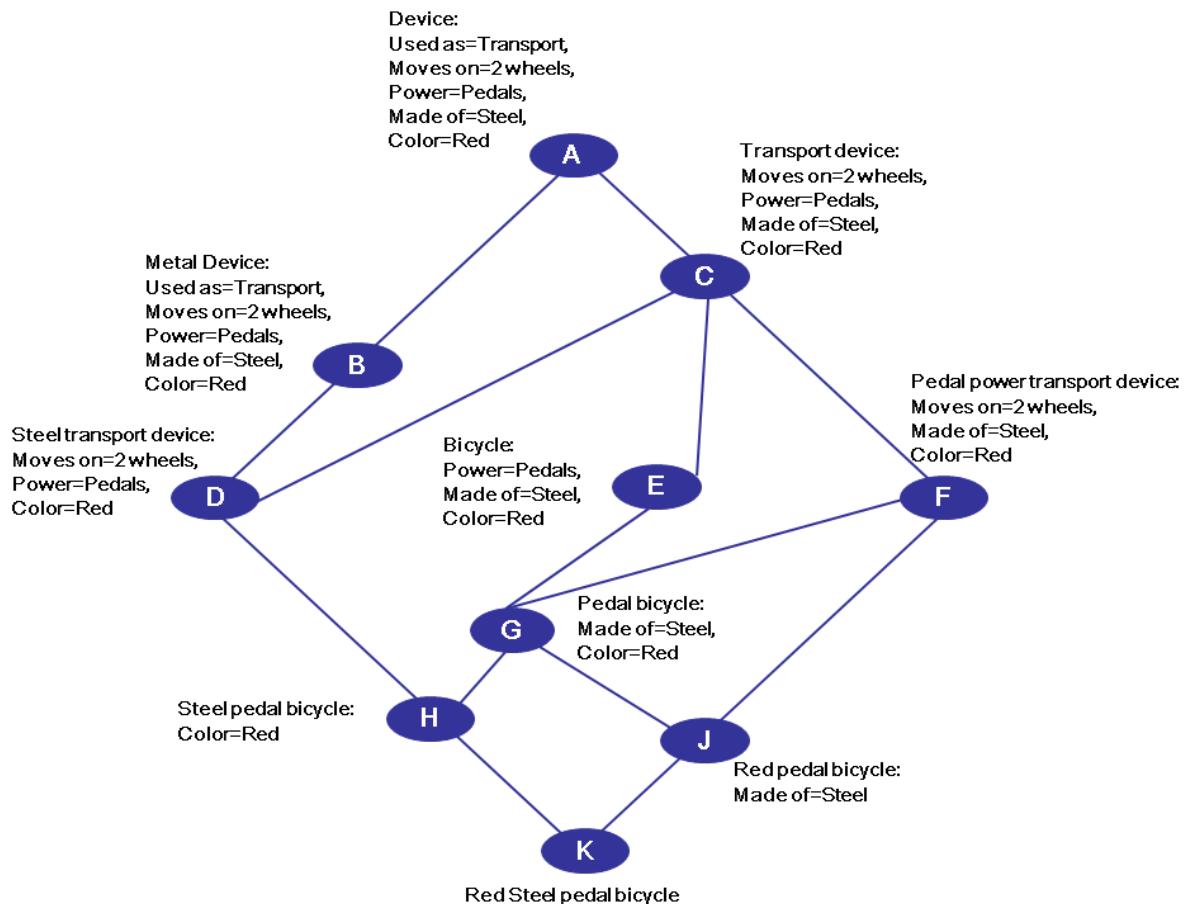


Figure 28: Alternative expressions that mean "red steel pedal bicycle"

Two rules limit the range of equivalent expressions:

Rule 1: It is not possible to transform a *primitive concept* into a *postcoordinated expression*.

- A *primitive concept* has a facet that is not represented by its *defining characteristics* and therefore any attempt to represent it in a *postcoordinated form* results in a loss of information.

This is illustrated by consideration of the definitions of the concept "bicycle" in [Table 14](#). The definition stated in the table is as follows:

| is a | = "Transport device", | Used as | = | Transport |, | Moves on | = | 2 wheels |, | Origin | = | Man made |

This definition would also apply to a horse-drawn cart or a trailer. Therefore the concept "bicycle" must be regarded as *primitive*. Recognizing this fact means that some of the apparently equivalent expressions in [Figure 28](#) cannot be computed as equivalent. Unless the *focus concept* is a *subtype* of "bicycle" it is not possible to compute that it is a kind of bicycle. This means that to create an equivalent expression it would be necessary to add | is a | = "Bicycle". This is shown in [Figure 29](#).

Examining these definitions, it is apparent that the characteristics shown in gray are redundant because they are part of the definition of "bicycle."

As a consequence of this rule, *primitive concepts* create the limits on the ability to transform an *expression* to a more post-coordinated form. An *expression* can be normalized until all the *concepts* referred to by the *expression* are *primitive*. An *expression* in which all the referenced *concept* are *primitive* is referred to as the *normal form*.

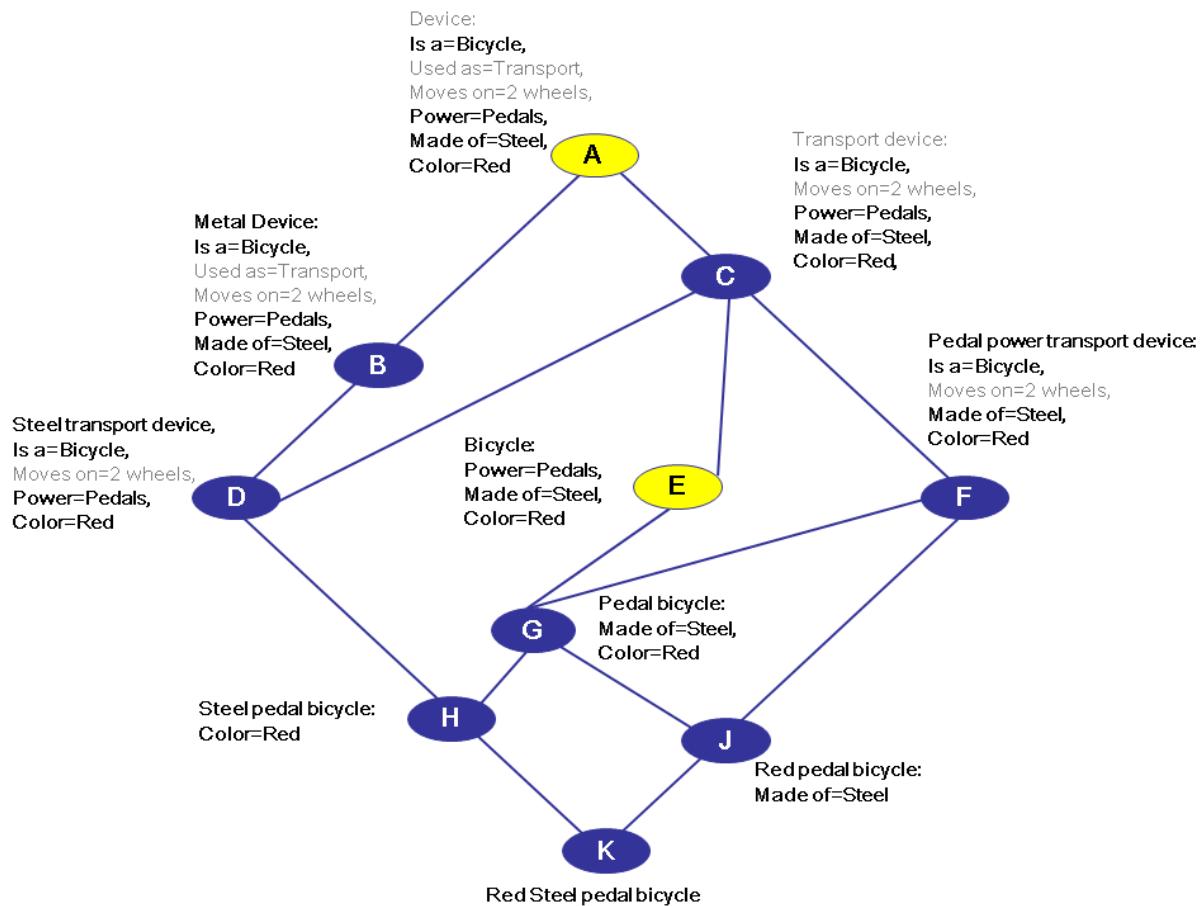


Figure 29: Expressions meaning "red steel pedal bicycle" with "bicycle"recognized as primitive

Rule 2: It is not possible to transform a *postcoordinated expression* into a fully *precoordinated concept* unless such a *concept* already exists in the released terminology.

This second rule is perhaps self-evident but it is stated because, like the first rule, it alters the available representations. If the *concept* "red steel pedal bicycle" was not available in a *precoordinated* form, there are two distinct *expressions* that are as *precoordinated* as possible (i.e. "steel pedal bicycle" + "color" = "red" and "red pedal bicycle" + | made of | = | steel |). This is illustrated in [Figure 30](#). In such cases there is no obvious reason to prefer one compared to the other.

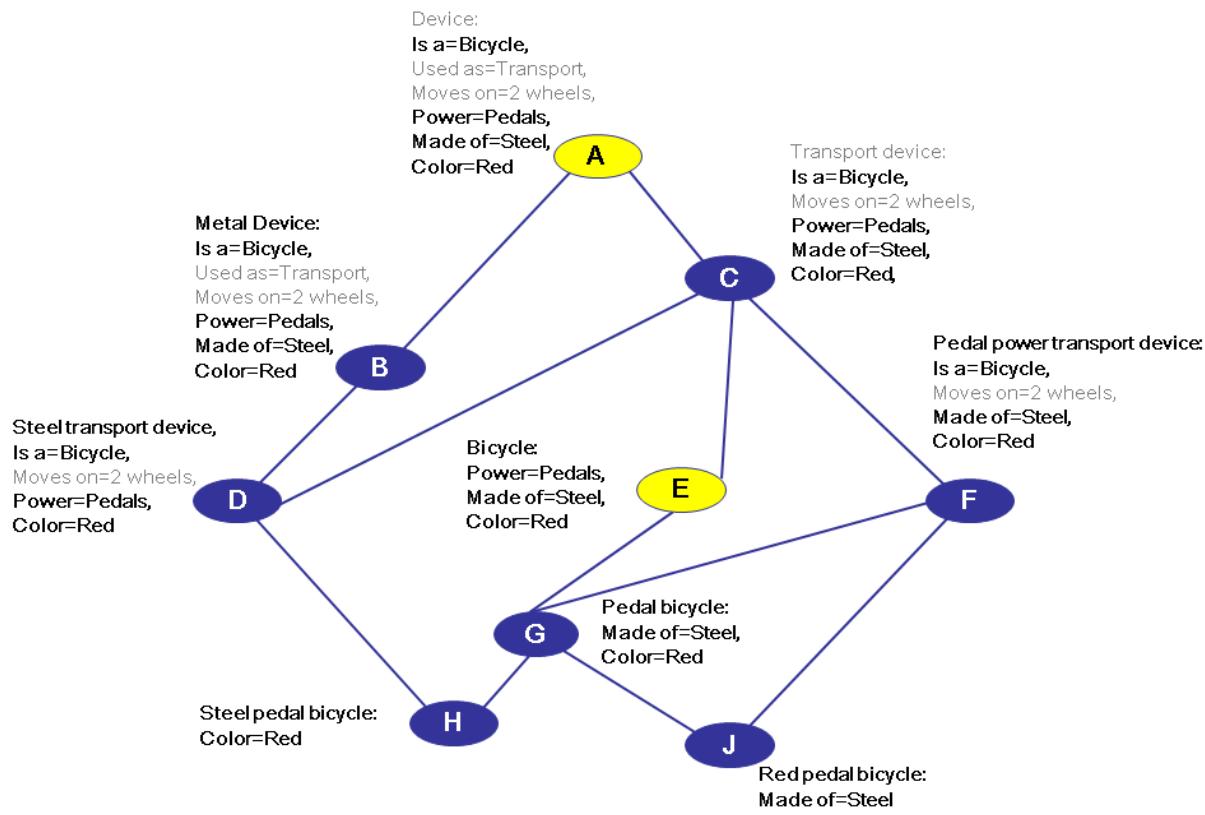


Figure 30: Possible expressions of "red steel pedal bicycle" without precoordinated option

4.3 Representational Forms



This section describes different ways in which *SNOMED CT components*, *derivatives* and *expression* can be represented. These representations include the files in which *SNOMED CT* is distributed as well as possible representations that may be used assist implementation or optimize particular functions.

4.3.1 Release Files



SNOMED CT is provided to licensees as a set of *release files*. The file naming conventions and the structure of these files is described in [Release File Specifications \(5\)](#) in a separate section of this guide. There are currently two distinct *Release Formats*:

- [Release Format 1 \(RF1\)](#): The specification in which *SNOMED CT* has been provided since its first release in 2002 (with a few minor amendments).
- [Release Format 2 \(RF2\)](#): Based on a draft trial specification that adds a range of significant enhancements.

4.3.2 Representing SNOMED CT identifiers



Components within *SNOMED Clinical Terms* are identified and referenced using numeric *Identifiers*. These *Identifiers* have the data type *SCTID* (*SNOMED CT Identifier*).

The *SCTID* data type is 64-bit *integer* which is allocated and represented in accordance with a set of rules. These rules enable each *Identifier* to refer unambiguously to a unique component. They also support separate partitions for allocation of *Identifiers* for particular types of component and *namespaces* that distinguish between different issuing organizations .

4.3.2.1 SCTID Data Type



The *SCTID* data type is a 64-bit positive *integer*.

When rendered as a string an *SCTID* must always be represented using decimal digits and the string rendering has a string the maximum permitted length of 18 digits and a minimum length of 6 digits.

 **Note:** Leading zeros are always omitted from the string rendering of an *SCTID*. For example the value "101291009" must not be rendered as "0101291009".

4.3.2.2 SCTID Representation



Each *SCTID* identifies a *SNOMED CT component*. The *Identifier* itself does not contain information related to the meaning of a *concept* or *description*. This means it is not possible to infer anything about the meaning of a *concept* from the numeric value of the *Identifier* or from the sequence of digits in that form the identifier. The meaning of a *concept* can be determined from *relationships* to other *concepts* and from associated *descriptions* that include human readable terms.

The *SCTID* does however have a structure which includes valuable information about the nature and source of the identified component and the validity of the *Identifier*. This structure supports the following features:

- *Check-digit* validation of the *Identifier*.
 - The *check-digit* is the final digit in the decimal rendering of the *Identifier*. This can be checked to minimize errors from transcription or incomplete copy-paste actions.
- Partitioning between *Identifiers* for different types of *SNOMED CT component*.
 - A two-digit *partition-identifier* distinguishes the *Identifiers* of different component types and prevents the same *Identifier* cannot be allocated to both a *concept* and a *description*. As a result, when an *SCTID* is read from a record or other resource, it is possible to determine whether it represents a *concept*, a *relationship* or a *description*, before searching for the identified component.
- Namespaces to separate component *Identifiers* originated by different organizations.
 - Organizations are only permitted to issue *Identifiers* which fall within a specified namespace of potential *Identifier* values. This prevents collisions between *Identifiers* issued by different organizations which would otherwise result in ambiguity and errors when sharing data.
 - There are two formats used for representing namespaces.
 - Short format in which *partition-identifiers* are reserved for an organization which is permitted to issue any valid *Identifiers* within the allocated partitions. The short format approach does not require a specific *namespace-identifier* and is only applicable to components originated and maintained by the *IHTSDO* as part of the *International Release* of *SNOMED CT*.
 - Long format in which the *partition-identifier* value indicates that a separate *namespace-identifier* is required to distinguish between components originated as part of an *Extensions* created by an appropriately authorized organization.

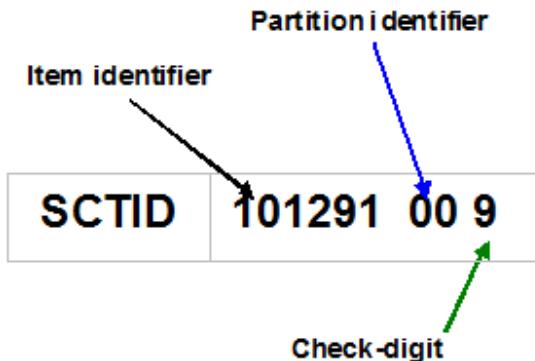


Figure 31: SCTID Short Format - Applicable to components originating from the International Release

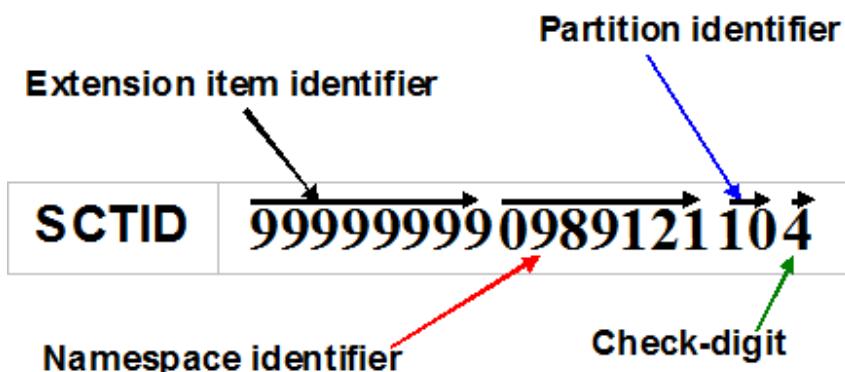


Figure 32: SCTID Long Format - Applicable to components originating from a SNOMED CT Extension

Note: The IHTSDO allocates *namespace-identifiers* to organizations such as *IHTSDO Members* and *Affiliates* to enable them to create content and or derivatives in an *Extension*. The *namespace-identifiers* enables unique *SCTIDs* to be issued by many organizations and allow each *SCTID* to be traced to an authorized originating organization .

4.3.2.3 SCTID Constraints



The permissible value for the *SCTIDs* are limited by the following rules:

- Only positive integer values that are greater than 10^5 and less than 10^{18} are permitted.
- The only valid *string* renderings of the *Identifier* value are *strings* of decimal digits (0-9), commencing with a non zero digit.
- The second and third digits from the right hand end of the *string* rendering of the *Identifier* must match one of the *partition-identifier values specified in this guide*.
- The rightmost digit of the *string* rendering is a *check-digit* and must match the value calculated using the specified *check-digit computation*.

Notes:

1. As a result of these rules, many 64-bit integers are not valid *SCTIDs*. The value limitations enable any valid *SCTID* to be stored in either a signed or unsigned 64-bit integer.
2. The rules also ensure that an *SCTID* can be distinguished from code from one of the antecedent code systems *Read Codes* (which are 4 or 5 characters in length) and legacy *Identifiers* from *SNOMED-RT* and its predecessors (which always start with a letter).

4.3.2.4 Check-digit



The final (units) digit of the *SCTID* is the *check-digit*. It is not envisaged that users will be routinely required to type *SCTID* values. However, the objective of the *check-digit* is to detect the commonest types of error that may occur due to typographical errors on those occasions where transcription or communication mechanisms may introduce error. Examples may include high-level development such as creating or modifying protocols or pre-specified queries.

An *SCTID* is checked by using the "Verhoeff check", which is a Dihedral D 5 Check. This detects a higher proportion of common typographical errors than either the IBM or Modulus 11 check. Unlike the Modulus 11 check it is effective on decimal strings longer than ten-digits. Furthermore its value can always be represented as a decimal digit without excluding any values.

See [Check-digit computation](#) for detailed information about the Verhoeff *check-digit* and sample program code.

4.3.2.5 Partition identifier



The penultimate two-digits of the *SCTID* (second and third from the right), are the *partition-identifier*.

The *partition-identifier* indicates the nature of the component identified. This allows the *Identifier* of a *Description* to be distinguished from the *Identifier* of a *Concept*.

The *partition-identifier* also indicates whether the *SCTID* contains a *namespace-identifier (long format)* or follows the *short format* applicable to *Identifiers of components* that originated in the *International Release*.

Identifiers of components that originated in the *International Release* of *SNOMED CT* have one of the following *partition-identifier* values:

Table 15: partition identifier Values for Short Format SCTIDs

PartitionId	Description
00	A <i>Concept</i>
01	A <i>Description</i>
02	A <i>Relationship</i>
<i>The values below are only valid in Release Format 1 they are not used in RF2.</i>	
03	A <i>Subset</i>
04	A <i>Cross~Map Set</i>
05	A <i>Cross~Map Target</i>

Identifiers of components that originated in an Extension have one of the following partition identifier values:

Table 16: partition identifier Values for Long Format SCTIDs

PartitionId	Description
10	A Concept
11	A Description
12	A Relationship
<i>The values below are only valid in Release Format 1 they are not used in RF2.</i>	
13	A Subset
14	A Cross~Map Set
15	A Cross~Map Target

All other *partition-identifier* values are reserved for future use.

4.3.2.6 Namespace-Identifier



If the *partition-identifier* indicates a long format *SCTID*, the seven-digits immediately to the left of the partition-digit are a *namespace-identifier*. The *namespace-identifier* is an integer value, left padded with '0's as necessary to ensure there are always seven digits in the value. The *namespace-identifier* does not hold meaning.

Each organization that is authorized to generate *SCTIDs* is allocated a *namespace-identifier* by the IHTSDO. Each allocated namespace is represented in the *Namespace Concept* metadata sub-hierarchy, released as part of the *International release* (see details in [The Namespace hierarchy](#)).

4.3.2.7 Item Identifier digits



The string of digits to the left of the *partition-identifier* (in a *short format SCTID*) or to the left of the *namespace-identifier* (in a *long format SCTID*) is referred to as the *item-Identifier*. These values are available to uniquely identify an individual entity within the specified partition or namespace. The same *item-Identifier* can be allocated in each partition of each namespace as the *SCTID* is rendered unique by the *partition-identifier* and the *namespace-identifier*.

For components in the *International Release* of *SNOMED CT*, *item-Identifiers* will usually be issued in the arbitrary order in which components are added to *SNOMED Clinical Terms*. However, due to management of the editing process the sequence of issued *item-Identifiers* may be discontinuous.

Caution: In all cases, the value of *item-Identifier* on its own is meaningless. The only way to determine the meaning of an *SCTID* is by looking up the complete value in an appropriate distribution file.

4.3.2.8 Example SNOMED CT identifiers



The following examples conform to the *SNOMED CT identifier* specification and illustrate a range of possible *Identifiers* within different partitions and namespaces.

SctId	Partition identifier	Check digit	Notes
100005	00 =Concept, using short format	5	The Item Identifier digits '100' are the lowest permitted value. Therefore this is the lowest SctId that can be allocated to a Concept.
100014	01 =Description, using short format	4	This is the lowest SctId that can be allocated to a Description.
100022	02 =Relationship, using short format	2	This is the lowest SctId that can be allocated to a Relationship.
1290023401004	00 =Concept, using short format	9	A valid SctId for a Concept.
1290023401015	01 =Description, using short format	5	A valid SctId for a Description.
9940000001029	02 =Relationship, using short format	9	A valid SctId for a Relationship.
10000001105	10 =Concept, using long format	5	A valid long format SctId for a Concept in the 0000001 namespace.
10989121108	10 =Concept, using long format	8	A valid long format SctId for a Concept in the 0989121 namespace.
1290989121103	10 =Concept, using long format	3	A valid long format SctId for a Concept in the 0989121 namespace.
1290000001117	11 =Description, using long format	7	A valid long format SctId for a Description in the 0000001 namespace.
9940000001126	12 =Relationship, using long format	6	A valid long format SctId for a Relationship in the 0000001 namespace.
999999990989121104	10 =Concept, using long format	4	The maximum valid SctId for a Concept in the 0989121 namespace.



The Namespace hierarchy

SNOMED CT core release files include metadata *Concepts* that represent each of the allocated *namespace-identifiers*. The *Concepts* representing the namespaces are arranged in a single parent hierarchy, as follows:

- 370136006 | Namespace concept |
 - 373872000 | Core Namespace |
 - | Extension Namespace A {} |
 - | Extension Namespace B {} |
 - | Extension Namespace D {} |
 - | Extension Namespace E {} |
 - | Extension Namespace C {} |

Figure 33: Hierarchy for: Namespace concept (namespace concept)

In the above hierarchy, | Extension Namespace A {} |, | Extension Namespace B {} | and | Extension Namespace C {} | are all child namespaces of the 373872000 | Core Namespace | (representing the *International edition* which does not have a *namespace-identifier*, and uses short format *SCTIDs* to identify *components*). Also, | Extension Namespace B {} | is the parent namespace of | Extension Namespace D {} | and | Extension Namespace E {} |.

Each *Namespace concept* may only have one parent *Namespace concept* in the 370136006 | Namespace concept | sub-hierarchy.

The namespace hierarchy is used to constrain which content can be promoted from one *Extension* to another without amending the *SCTID*. Content may be moved (without amendment of *SCTID*) from an *Extension* released by the owner of a child namespace to an *Extension* released by the owner of a parent (or ancestor) namespace, as described by the |370136006 | Namespace concept | sub-hierarchy.

👉 Examples:

1. A *concept* with an *SCTID* that includes | Extension Namespace D {} | may be moved to the *Extension* maintained by the owner of | Extension Namespace B {} | without changing its *SCTID*, because this is a parent of the originating namespace.
2. A *concept* with an *SCTID* that includes | Extension Namespace D {} | must not be moved to the *Extension* maintained by the owner of | Extension Namespace C {} | because this is not parent (or ancestor) of the originating namespace. Therefore, to make this move the original *concept* must be inactivated and replaced by a new component with a new *SCTID* in target namespace.
3. Any *concept* may be moved from any *Extension* to the *International Release* (subject only to formal acceptance that is a valid addition for international use).

Namespace concepts have the following characteristics:

- They are *subtypes* (either children or *descendants*) of 370136006 | Namespace concept |.
- The *Fully Specified Name* of each *Concept* has the form “ *Extension Namespace {nnnnnnn} (namespace concept)*” – where nnnnnnn is the seven digit *Namespace-Identifier*.
- A *Synonym* associated with each *Concept* has the form “ *Extension Namespace nnnnnnn*”
- Where appropriate further *Synonyms* may be included to identify the nature of the responsible organization.

When requesting a *namespace-identifier* from *IHTSDO*, there will be a facility to optionally specify a parent *Namespace-identifier* for the new namespace.

To specify a parent namespace for an existing *namespace-identifier*, please contact info@ihtsdo.org with details of your existing *namespace-identifier* and its proposed parent *namespace-identifier*.

Caution: Once a *namespace-identifier* has been allocated a parent *namespace-identifier* in this hierarchy, further changes to this hierarchical *Relationship* are not permitted. This restriction is imposed to avoid changes that would undermine traceability of moves between namespaces.

4.3.3 Representing Extensions



4.3.3.1 Extension Tables - Structure



Extensions use the same table structure as the Concepts, Descriptions, Relationships, and Reference Sets tables defined in those respective sections of this manual. These tables have the same structure or schema as the core tables but are in separate files.

When packaged, extension file names should follow the conventions defined by the IHTSDO. For more information, refer to the document *SNOMED CT File Naming Convention*.

4.3.3.2 Specification for Namespace within the SCTID



The identifiers assigned to all components that originated as part of an extension include a *namespace-identifier* (see [Representing SNOMED CT Identifiers](#)). This means that the sets of *Identifiers* available to each organization authorized to issue components are distinct, which ensures that the same *Identifier* cannot be issued by two different organizations.

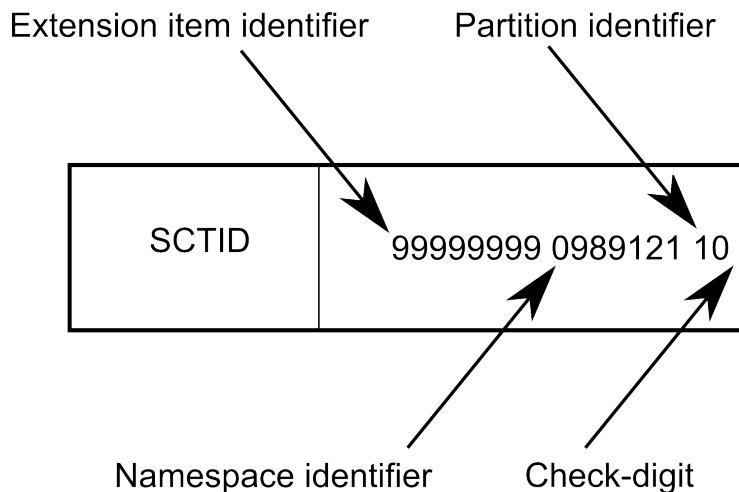


Figure 34: SCTID for a concept originated as part of an Extension

All Extension components (rows) originated by an organization must use the Namespace Identifier assigned to that organization. Namespace Identifiers are issued by the IHTSDO so that the Namespaces remain unique between organizations. Allocated Namespaces are recorded as Concepts in the | SNOMED CT Model Component | hierarchy ("special concepts" hierarchy in RF1) when they are issued to an organization.

Namespaces serve three Roles:

- Preventing collision or reuse of SNOMED CT identifiers;
- Indicating the origin of a component. In RF2 responsibility for maintenance is tracked only by the ModuleId, this is the field that should be used to avoid potential risk of two organizations make conflicting changes to the same component.
- Indicating the source for information about a concept - relevant for Extension Concepts that are not directly available in a particular system.

All Extension components (rows) should use the appropriate partition-identifiers for Extensions. This ensures that components of the SNOMED CT International Release can be distinguished from components that are part of an Extension.

 **Note:** Components that originate as part of the *International Release* do not have a *namespace* field and are distinguished instead by *partition-identifier* values that are specific to *International Release*.

 **Note:** Each organization can assign the *item-Identifier* portion of the *SCTID* in any way within its *Namespace*. If there is a need to allocate part of the development process to a subdivision within an organization, they may be allocated a set or range of *item-identifiers* that have not yet been used or allocated within that *Namespace*. The authorized organization must ensure that it tracks and manages all such allocations in a way that avoids any risk of reuse of the same *SCTID*.

4.3.3.3 Namespace allocation



Namespace-identifiers are allocated by the *IHTSDO* to licensed organizations. The *IHTSDO* is under no obligation to allocate a *namespace* to any organization and makes these allocations at its discretion.

Allocation of a *namespace* does not imply any endorsement of the reputation of an organization nor to the quality or fitness for purpose of any *Extensions* created by that organization. Users and/or vendors incorporating *Extensions* into their application do so at their own risk and should satisfy themselves with the reputation of the responsible organization and the quality of the *Extensions* so incorporated.

4.3.3.3.1 Namespace Allocation Policy/Regulation



Title: Namespace Allocation Policy/Regulation

Effective Date: August 4, 2009 **Owner :** Management Board

Date Last Reviewed: October 14, 2009 **Date Last Revised :** October 14, 2009

4.3.3.3.1.1 Regulation Statement



IHTSDO will allocate *SNOMED CT Namespace Identifiers* upon written request from a Member or an Affiliate in accordance with the procedures outlined below. The *IHTSDO* will also maintain and publish a register of *Namespace Identifiers* issued.

Section 9 of the Articles of Association provides the starting point for the *Namespace Allocation Policy*. It states that:

- 9.1 Only the Association may issue *Namespace Identifiers*.
- 9.2 The Association shall, upon written request from a Member or an Affiliate in accordance with such procedures as the Association may prescribe by Regulations, issue one or more *Namespace Identifiers* to the Member or Affiliate. The Association shall not unreasonably refuse to issue a *Namespace Identifier* to a Member or an Affiliate.
- 9.3 The Association shall be responsible for ensuring that each *Namespace Identifier* is only issued to a single Member or Affiliate.

In addition, section 7.1.7 states that "An Affiliate may not create any Standards-Based Third Party *Extension* or any Standards-Based *Derivative* from the Member's National *Extensions* unless that Affiliate has been issued with a *Namespace Identifier*."

4.3.3.3.1.2 Definitions



Affiliate: An Affiliate of *IHTSDO* in accordance with *IHTSDO's Articles of Association*, i.e. a person or organization to which the *International Release* of *SNOMED CT* (whether on its own or as part of a Member's *National Release* of *SNOMED CT*) is distributed or otherwise made available under the *Articles of Association*.

Namespace Identifier: A code or that part of a code that identifies the organization responsible for creating and maintaining a standards-based *extension* or a standards-based *derivative*. It is an element of *SNOMED CT concept Identifiers*.

4.3.3.3.1.3 Context



Namespace Identifiers are 7-digit numbers that *IHTSDO* issues to those who create *extensions* to *SNOMED CT*, such as national *extensions*. *Namespace identifiers* ensure that it is clear who developed

and maintains particular customized terminology. They also ensure that terminology in *SNOMED CT extensions* has unique *Identifiers* but a common structure, which facilitates application development and the creation of *Reference Sets*. There is a defined process for management of *Namespace Identifiers* when terminology is moved between *extensions* or from an *extension* into the *International Release*.

It should be noted that this policy covers the technical mechanism to allocate *Namespace Identifiers* in order to be able to identify the source of content, prevent collisions in terminology that would affect interoperability, and achieve similar goals. It does not cover whether or not particular types of content, including *extensions* and *derivatives*, can be used in a given context. This may be the subject of national policies, guidelines, or other documents. Requesters of a *Namespace Identifier* are encouraged to review *SNOMED CT International Release* documentation and to consult with *IHTSDO Members* in countries in which deployment of any content developed in the *Namespace* is planned for additional guidance, policy, and/or process documents which may be relevant.

4.3.3.3.1.4 Procedures

4.3.3.3.1.4.1 Informing the Community of Practice :



IHTSDO will inform the Community of Practice about the process for requesting a *Namespace Identifier*.

- A copy of this regulation will be posted to the *IHTSDO* website.
- Instructions for requesting a *Namespace Identifier*, including the form for making such a request, will be posted to an appropriate location on the *IHTSDO* website (e.g. the Frequently Asked Question or "How do I?" pages).
- Confirm which Members would like to be notified when one of their Affiliates requests a *Namespace*, i.e. an Affiliate listing an address in the jurisdiction in question on their *Namespace Identifier* Application Form and/or who identified that they received their Affiliate License through that jurisdiction.

4.3.3.3.1.4.2 Requesting & Granting a Namespace:



The Association shall, upon written request from a Member or an Affiliate in accordance with such procedures as the Association may prescribe by Regulations, issue one or more *Namespace Identifiers* to the Member or Affiliate. The Association shall not unreasonably refuse to issue a *Namespace Identifier* to a Member or an Affiliate.

- To request a *Namespace Identifier*, individuals/ organizations should complete and submit a copy of the *Namespace Identifier* Application Form by email to support@ihtsdo.org.
- *IHTSDO* or its designated support organization should verify that the requester is either:
 - An *IHTSDO Member*
 - An individual or organization who holds a valid AffiliateLicense
 - An individual or organization who does not fall into the above categories but whose application is approved in writing by the *IHTSDO*
- If the conditions above apply, *IHTSDO* or its designated support organization should issue a unique *Namespace Identifier* to the requester if:
 - The request is from an *IHTSDO Member*,
 - The request is from the holder of a valid Affiliate License , is for a single *Namespace Identifier*, and the requester has not already been issued a *Namespace Identifier*; OR;
 - *IHTSDO's CEO* approves the request in writing.
- The issuance of the *Namespace Identifier* should be confirmed in writing with the requester, along with a link to this policy and a reminder that they will be contacted annually to reconfirm their contact information and potentially provide additional information to be published in the register. Requesters should also be provided with Member contact information and the recommendation that they should contact relevant Members to obtain any additional national guidance, policy, and process documents which may be relevant.
- The relevant Member should be informed if they have requested to be notified when an Affiliate from their jurisdiction requests a *Namespace*, i.e. an Affiliate listing an address in the jurisdiction in question on their

Namespace Identifier Application Form and/or who identified that they received their Affiliate License through that jurisdiction.

4.3.3.3.1.4.3 Format of Namespace Identifiers:



- *Namespace Identifiers* are 7 digit numeric codes;
- *Namespace Identifiers* are issued in sequence, unique, and not re-used.

4.3.3.3.1.4.4 Maintaining and Publishing a Namespace Register:



IHTSDO will maintain and publish an up-to-date *Namespace Register*.

- When *Namespace Identifiers* are allocated, a record of the number of that *Identifier*, the date of issuance, the body from which the Affiliate License was obtained (if applicable), and the name and contact information of the individual/organization to which it was issued will be added to the *Namespace Register*.
- On an annual basis, *IHTSDO* or its designated support organization will contact all those to whom *Namespace Identifiers* have been issued by email to confirm contact information. A reminder will be sent after 30 days if a response has not been received. The year in which a confirmation of *current* contact information was last received will also appear in the *Namespace Register*.
- *IHTSDO* reserves the right to make a *Namespace Identifier* *inactive* for *current* and future use (i.e. it cannot be used for newly-created *concepts* from that point onward) if the individual or organization to which it was issued cannot be contacted after three attempts. This *status* will be noted in the *Namespace Register* and the individual or organization to which the *Namespace Identifier* was issued will be notified accordingly.
- *IHTSDO* also reserves the right to make a *Namespace Identifier* *inactive* if (1) it is requested to do so by the organization to which the *Namespace Identifier* was issued, (2) the organization to which the *Namespace Identifier* was issued is involved in a merger or acquisition with another organization to which a *Namespace Identifier* has been issued, or (3) it receives a written complaint about the use of that *Namespace* that, upon investigation, it determines to be well-founded, according to the protocol for material breaches and termination of Affiliate Licenses identified in clause 5.2 of the Affiliate License .
- The *Namespace Register* will be published with each version of the *SNOMED CT International Release*. In the future, *IHTSDO* reserves the right to also publish the *NamespaceRegister* on the *IHTSDO* website.

4.3.3.3.1.5 References



- *IHTSDO Articles of Association* ;
- *SNOMED CT Technical Implementation Guide* ;
- *IHTSDO Namespace Identifier Application Form* .

4.3.3.3.1.6 Document Control



This policy was approved by the *IHTSDO Management Board* on August 4, 2009 and is subject to regular review according to *IHTSDO's* policy review processes. Key stakeholders include the Technical Committee, the Implementation and Innovation Committee, the Member Forum, and the Affiliate Forum.

4.3.3.4 Component Guidelines



Descriptions that are part of an *Extension* can refer to either a *Concept* that is part of that *Extension*, a *Concept* that is part of another *Extension*, or an *International Release Concept*.

Relationships that are part of an *Extension* can relate two *Concepts* in the *Extension* or two *Concepts* in different *Extensions*. The *relationship* can also relate the *extension Concept* to an *International Release Concept*-- that is, *sourcelId* is in the *extension* and *destinationId* is in the *International Release*.

4.3.3.5 Content of Extensions



The *components* in an *Extension* have the same structure as *International Release components* of *SNOMED CT*.

SNOMED CT International Release is not dependent on availability of any *Extension*. However, all *SNOMED CT Extensions* are dependent on the *SNOMED CT International Release*. Some *Extensions* may also be

dependent on other *Extensions*. Dependencies between *Extensions* must be declared and must not be circular.

The following rules apply to dependencies between *components* and *derivatives* in *Extensions*.

 **Note:** In these rules,

- *Containing-Extension*, refers to the *Extension* that contains the named *component* or *derivative*.
- *Dependee-Extension*, refers to another *Extension* on which the *Containing-Extension* is dependent.

1. Every Concept in an *Extension*:

- Must be a *subtype descendant* of an *International Release Concept*.
- This descent may be indirect, passing through *Concepts* in either the *Containing-Extension* or a *Dependee-Extension*.

2. Every *Description* in an *Extension*:

- Must apply to a *Concept*, in the one of the following *Namespaces*: the *Containing-Extension*, the *International Release* or a *Dependee-Extension*.

3. Every defining *Relationship* in an *Extension*:

- Must define a *sourceld* which refers to a *Concept* in the *Containing-Extension*.
 - In exceptional circumstances, an *Extension* may add additional defining attributes to a *Concept* in the *International Release* or a *Dependee-Extension*.
- Must have *typeld* which refers to a *Concept*, in the one of the following *Namespaces*: the *Containing-Extension*, the *International Release*, or a *Dependee-Extension*.
- Must have a *destinationld* which refers to a *Concept*, in the one of the following *Namespaces*: the *Containing-Extension*, the *International Release*, or a *Dependee-Extension*.

4. A *Reference Set* in an *Extension*:

- May include references to *components* and *derivatives* in any of the following *Namespaces*: the *Containing-Extension*, the *International Release*, or a *Dependee-Extension*.

5. The enumerated values used in *RF2 components* that form part of an *Extension* must be all be represented by metadata *Concepts* that are present in one of the following *Namespaces*: the *Containing-Extension*, the *International Release*, or a *Dependee-Extension*.

The following additional rules are only relevant to *Extensions* represented using *Release Format 1*:

• *Reference Sets* in an *Extension*:

- May include as its members *Components* from the following *Namespaces*: the *Containing-Extension*, the *International Release*, or a *Dependee-Extension*.
- May refer to other *Reference Sets* in the following *Namespaces*: the *Containing-Extension*, the *International Release*, or a *Dependee-Extension*.
- May provide maps for *Concepts* from the following *Namespaces*: the *Containing-Extension*, the *International Release*, or a *Dependee-Extension*.

4.3.3.6 Transfer of Responsibility between Organizations



When the need arises to transfer *components* (*Concepts*, *Descriptions*, *Relationships*) from the *International Release* content to an *Extension*, from an *Extension* to the *International Release*, or from one *Extension* to another, conversation and coordination between the sending and receiving organizations is needed. In some cases, entire tables may be transferred - not just individual components.

It should be noted that the transfer of *components* among *Extensions*, or between an *Extension* and the *International Release*, is subject to the terms of the *IHTSDO Affiliate License* and, within an *IHTSDO Member country*, may also be subject to the *terms* of that Member's *SNOMED CT* national license .

Examples of transfers include:

- From the *SNOMED CT International Release* to an organization responsible for an *Extension*:
 - This occurs if a decision is made that some *Concepts* in the *International Release* are specific to a *Realm* or domain or interest for which another organization has been allocated responsibility:
 - For example, this applies to UK specific drugs and UK specific administrative *Concepts* which are maintained by the *UK NHS*.
- From an organization responsible for an *Extension* to the *International Release*:
 - This occurs if an organization recognizes that some of its *Extension* content belongs in the *International Release* as it has general applicability;
 - It also occurs if an organization hands over responsibility for its entire *Extension* to the *IHTSDO*.
- From one organization responsible for an *Extension* to another organization :
 - This occurs if one organization recognizes that some of its *Extension* content belongs in a domain managed by another organization ;
 - It also occurs if an organization hands over responsibility for its entire *Extension* to another organization

There are three types of transfer of responsibility:

- Transfer of an entire *Extension* (i.e. all *components* ever issued with an *SCTID* in a given *Namespace*. from one organization to another organization):
 - This is a straight forward process. All that happens is that another organization assumes responsibility for the original *Namespace-identifier*. There is no need for detailed tracking of individual *components*.
- Transfer of one or more *components* from an *Extension* to the *International Release* or to a "parent"*Extension*:
 - As a result of revision to guidance issued by the *IHTSDO* in 2011, these transfers can be made without changing the *Identifier* of the *component* provided that the RF2 format *moduleId* field is used to denote that the *component* is now being issued as part of a different module.
- Transfer of one or more *components* between other *Extensions* or from the *International Release* to any *Extension*:
 - In this case, the *Namespace* is not transferred and thus, to fulfill the roles of the *Namespace-Identifier*, the *component* must be assigned new *SCTIDs* in the *Namespace* of the newly responsible organization :
 - The previous instances of these *components* are withdrawn from *current* use with the *Status* value *Moved Elsewhere*;
 - Appropriate *Relationships* point to replacement *components* in the new *Namespace*.

The transfer of responsibility depends on the release schedules of the organizations involved. Often the original source organization will be aware of an intended move before the target organization has accepted responsibility and released the *component*. To facilitate this, an interim *Status* value *Pending Move* is applied to *components* that are being moved to another *Namespace* but are intended for *active* use until their replacements are found in the target *Namespace*.

To provide continuity for a *Concept* if responsibility is transferred, the *Concept Status* and history are coordinated as follows:

Table 17: Processing Transfers between Extensions

	Sending Organization	Receiving Organization ⁴
Start State	Concept A <i>Concept Status = Current</i>	
Agreement to transfer responsibility	Concept A <i>Concept Status = Move Pending</i>	
Responsibility Transferred	Concept A <i>Concept Status = Moved Elsewhere</i> History Files = Concept A "Status Change" to "Moved Elsewhere" History Files = Concept A "Moved to" Namespace 9999999 in this release 👉 Note: Namespace 9999999 is recorded as a "Special Concept" in the SNOMED CT Concepts File. Therefore, the Sending Organization can track the organization to which the concept has moved, even if the new Concept Identifier is not yet assigned.	Concept B (Concept B is a new Concept Identifier using namespace = 9999999) <i>Concept Status = Current</i> History Files = Concept B "Added" History Files = Concept B "Moved from" Concept A 👉 Note: The Receiving Organization can record the Concept Identifier previously used for the concept. 👉 Note: If the Receiving Organization is the IHTSDO, the namespace would be the International Release namespace rather than the example used of namespace = 9999999
End State	Concept A is <i>Inactive</i> <i>Concept Status = Moved Elsewhere</i> History relates Concept A to the Receiving Organization by use of the Namespace Identifier	Concept B is <i>active</i> <i>Concept Status = Current</i> History relates Concept B back to Concept A

4.3.3.7 Released Extensions



The following extensions are included in the *International Release* of SNOMED CT from the IHTSDO. As with any extension, their content may not be suitable for use everywhere, and users should consult with their National Release Center for information regarding the use of extension content within an IHTSDO Member country.

⁴ Assume assigned namespace = 9999999

Table 18: Released Extensions

Extension	Distribution	Extension Contents
U.S. Drug Extension	International Release	<p>Actual manufactured drugs approved for distribution in the United States at the "actual medicinal product" (AMP) level. The AMP is a syntactic <i>normal form</i> consisting of:</p> <ul style="list-style-type: none"> • Name (Proprietary); • Strength; • Dosage Form. <p>All AMPs relate to "virtual medicinal product" (VMP) concepts in the <i>SNOMED CT Core</i>. All AMPs include the "has active ingredient" relationship where the <i>active ingredient</i> is a substance in the <i>SNOMED CT Core</i>.</p>

4.3.4 Representational Forms for Expressions



4.3.4.1 SNOMED CT compositional grammar



The *SNOMED* Composition Grammar is a lightweight syntax for representation of *SNOMED CT* expressions. It is has been proven to be both human readable and machine parsable.

4.3.4.1.1 Background

4.3.4.1.1.1 Prior versions and status of revision



The *SNOMED* Composition Grammar was initial specified as part of the document "*SNOMED Clinical Terms Abstract Logical Models and Representational Forms, External Draft for Comment Version*". This was used extensively and was proven to be both human readable and machine parsable.

The current specification which has now been adopted as an *IHTSDO* Standard, follows the prior version in most details. It includes the following enhancements:

1. The syntax of the grammar specification is now Augmented Backus-Naur Form (ABNF)⁵ which provides a formal standards-based reference for the grammar's structure.
2. Unnecessary whitespace designators, <ws>, were removed from several places in the grammar.
3. The maximum length constraint for *SNOMED Clinical Terms Identifiers* (SCTIDs) is added to this grammar. SCTIDs consist of sequences of digits, from a minimum of 6 to a maximum of 18 digits in length.
4. The hex code for carriage return (CR) was incorrectly given as '0C' in the previous version. It is corrected to '0D' in this version.
5. Detailed character encoding information for *UTF-8* is added.
6. The definition of *term* has been amended to allow correct parsing by the APG parser generator.

4.3.4.1.1.2 Compositional Grammar and the HL7 Code data type



The *SNOMED CT* compositional grammar allows *SNOMED CT* expressions to be represented as a text string that can be carried in *HL7* version 3 messages, in the *Code* data type. In particular, the grammar is intended to replace the *qualifier* mechanism that formerly was in the *HL7 Concept Descriptor* data type (CD data type), and which was removed in the *HL7* version 3 data types *Release 2*.

In May 2008, the *HL7* Version 3 Standard "Data Types - Abstract Specification, Release 2" was released for Normative Ballot 2.

⁵ ABNF as defined by Internet Standard 68, RFC 5234

This revised standard defined what can be carried in the *Code* data type as "the plain code symbol defined by the code system, or an *expression* in a syntax defined by the code system which describes the *concept*."

The following details are quoted from the *HL7 Version 3 Standard: Data Types - Abstract Specification, Release 2, Normative Ballot 2 - May 2008 (HL7 V3 DT R2)*, section 4.5.1 "Code (code): ST.SIMPLE":⁶

Table 19: Code definition from HL7 Data Types Release 2

Code (code) :*ST.SIMPLE*

Definition: The plain code symbol defined by the code system, ***or an expression in a syntax defined by the code system which describes the concept.*** (emphasis added)

If provided, the code SHALL be an exact match to a plain code symbol or *expression* defined by the code System. If the code system defines a code or *expression* that includes whitespace, the code SHALL include the whitespace.

An *expression* can only be used where the code System either defines an *expression* syntax, or there is a generally accepted syntax for the code System. (emphasis added)

The syntax described herein is intended to satisfy the need for a "syntax defined by the code system" as stated above, when the "code System" is *SNOMED CT*.

⁶ http://www.hl7.org/v3ballot/html/infrastructure/datatypes_r2/datatypes_r2.htm

4.3.4.1.2 Compositional grammar: Normative specification



Table 20: ABNF definition of the SNOMED CT compositional grammar

```

expression = concept *( "+" concept ) [ ":" ws refinements ]
concept = ws conceptId ws ["|" ws term ws "|" ws]
conceptId = sctId
term = 1*nonwsnonpipe *( 1*SP 1*nonwsnonpipe )
refinements = ( attributeSet *attributeGroup ) / 1*attributeGroup
attributeGroup = "{" attributeSet "}" ws
attributeSet = attribute *( "," attribute)
attribute = attributeName "=" attributeValue
attributeName = ws attributeNameId ws ["|" ws term ws "|" ws]
attributeValue = concept / ( ws "(" expression ")" ws)
attributeNameId = sctId
sctId = digitNonZero 5*17( digit )
ws = *( SP / HTAB / CR / LF ) ; white space
SP = %x20
HTAB = %x09
CR = %x0D
LF = %x0A
digit = %x30-39
digitNonZero = %x31-39 ; digits 1 through 9, but excluding 0
nonwsnonpipe = %x21-7B / %x7D-7E / UTF8-2 / UTF8-3 / UTF8-4
UTF8-2 = %xC2-DF UTF8-tail
UTF8-3 = %xE0 %xA0-BF UTF8-tail / %xE1-EC 2( UTF8-tail ) /
%xED %x80-9F UTF8-tail / %xEE-EF 2( UTF8-tail )
UTF8-4 = %xF0 %x90-BF 2( UTF8-tail ) / %xF1-F3 3( UTF8-tail ) /
%xF4 %x80-8F 2( UTF8-tail )
UTF8-tail = %x80-BF

```

4.3.4.1.3 Informative comments



Table 21: BNF representation of Compositional Grammar (detail)

```
Expression = concept *( "+" concept ) [ ":" ws refinements ]
```

	<p>An <i>expression</i> supports combinations of one or more <i>concepts</i> optionally refined by a set of <i>refinements</i>. The meaning of the <i>expression</i> is a <i>subtype</i> of all the <i>concepts</i> constrained by the set of <i>refinements</i>.</p> <p>Note that where there is a requirement for multiple separately qualified <i>concepts</i> to be present these are expressed in <i>attribute groups</i> within a <i>refinement</i> of a general <i>concept</i> such as "situation with explicit context".</p>
<i>concept</i> = ws conceptId ws [" " ws <i>term</i> ws " " ws]	
	<p>A <i>concept</i> is represented by a <i>conceptId</i> optionally followed by a <i>term</i> enclosed by a pair of " " characters.</p> <p>Whitespace before or after the <i>conceptId</i> is ignored as is any whitespace between the initial " " characters and the first non-whitespace character in the <i>term</i> or between the last non-whitespace character and before second " " character.</p>
<i>conceptId</i> = sctId	
	<p>The <i>conceptId</i> must be a valid <i>SNOMED CT identifier</i> for a <i>concept</i>. The initial digit may not be zero. The smallest number of digits is six, and the maximum is 18.</p>
<i>term</i> = 1*nonwsnonpipe *(1*SP 1*nonwsnonpipe)	
	<p>The <i>term</i> must be the <i>term</i> from a <i>SNOMED CT description</i> that is associated with the <i>concept</i> identified by the preceding <i>concept identifier</i>. For example, the <i>term</i> could be the preferred <i>description</i>, or the preferred <i>description</i> associated with a particular translation. The <i>term</i> may include valid <i>UTF-8</i> characters except for the pipe " " character⁷. The <i>term</i> begins with the first non-whitespace character following the starting " " character and ends with the last non-whitespace character preceding the next " " character.</p>
<i>refinements</i> = (attributeSet *attributeGroup) / 1*attributeGroup	
	<p>A <i>refinement</i> contains all the grouped and ungrouped attributes that refine the meaning of the containing <i>expression</i>. The ungrouped attributes, if any, are all listed first, followed by all the grouped attributes.</p>
<i>attributeGroup</i> = { " attributeSet " } ws	
	<p>An <i>attribute group</i> contains a collection of attributes that operate together as part of the <i>refinement</i> of the containing <i>expression</i>.</p>
<i>attributeSet</i> = attribute *("," attribute)	
	<p>An attribute set contains one or more <i>attribute name</i>-value pairs expressing <i>refinements</i>. They are separated by commas.</p>

⁷ The specification for term should be comparable with the specification for the Concepts.FullySpecifiedName and Descriptions. Term fields in the release table structure (as described in SNOMED Clinical Terms Technical Reference Guide, July 2008, IHTSDO). The non-pipe constraint adds greater stringency to the Compositional Grammar specification.

attribute= attributeName "=" attributeValue	
	An <i>attribute name</i> -value pair expressing a single <i>refinement</i> of the containing <i>expression</i> .
attributeName= ws attributeNameld ws [" " ws <i>term</i> ws " " ws]	
	The name (or <i>relationship type</i>) of an attribute to which a value is applied to refine the meaning of a containing expression. The <i>attribute name</i> is represented by an appropriate <i>conceptId</i> optionally followed by a <i>term</i> enclosed by a pair of " " characters. Whitespace before or after the <i>conceptId</i> is ignored as is any whitespace between the initial " " characters and the first non-whitespace character in the <i>term</i> or between the last non-whitespace character and before second " " character.
attributeValue= concept / (ws "(" expression ")" ws)	
	A <i>concept</i> or <i>expression</i> representing the value of a named attribute which refines the meaning of a containing expression. If an <i>expression</i> is used this must be enclosed in brackets.
attributeNameld = sctId	
	The <i>attribute name id</i> must be the <i>conceptId</i> for a <i>concept</i> that is a <i>subtype descendant</i> of the SNOMED CT concept "attribute".
sctId = digitNonZero 5*17(digit)	
	A <i>n sctId</i> is used for an attribute id or a <i>concept id</i> . The initial digit may not be zero. The smallest number of digits is six, and the maximum is 18.
ws= *(SP HTAB CR LF)	
	Whitespace characters (space, tab, linefeed and carriage return) are ignored everywhere in the <i>expression</i> except: <ol style="list-style-type: none">1. Whitespace within a <i>conceptId</i> or <i>attributeNameld</i> is an error. 👉 Note: Whitespace before or after the last digit of a valid <i>Identifier</i> is ignored.2. Whitespace within a <i>term</i> is treated as a significant character of the <i>term</i>. 👉 Note: Whitespace before the first or after the last non-whitespace character of a <i>term</i> is ignored
nonwsnonpipe= %x21-7B / %x7D-7E / UTF8-2 / UTF8-3 / UTF8-4	
	Non whitespace includes printable ASCII characters (these are also valid UTF8 characters encoded as one octet) and also includes all UTF8 characters encoded as 2- 3- or 4-octet sequences. It excludes space (which is %x20) and the pipe character " " (which is %x7C), and excludes CR, LF, HTAB and other ASCII control codes. SeeRFC 3629 (UTF-8, a transformation format of ISO 10646 authored by the Network Working Group).

digitNonZero= %x31-39	
	The first character of a <i>concept identifier</i> is constrained to a digit other than zero.
digit= %x30-39	
	Any digit 0 through 9

4.3.4.1.4 Examples of Grammar



The following examples build on each other and in complexity. They are primarily aimed at demonstrating the syntax of the *expression* grammar, although its meaning is also discussed in a number of places:

An *expression* may consist of a single *concept*, followed by a *description* associated with that *concept*. Which particular *description* to use is not mandated, but as a general rule, it may be preferable to use the *preferred term* in any particular *dialect* to achieve some level of consistency. However, such guidance is not strictly in the scope of this guide, and may be given elsewhere.

297186008 | motorcycle accident |

The syntax does not require a *description* to be associated with a particular *concept*, so the following is also a valid *expression*:

297186008

Two or more *concepts* may be combined to form a new *concept* by joining them with the "+" symbol. The resultant *expression* is the *child* of each of the *concepts* in the *expression*. The resultant *expression* below IS AN accident caused by a blizzard and also | is a | motorcycle accident.

217724009 | accident caused by blizzard | +297186008 | motorcycle accident |

Although not stipulated by the syntax, note that two *concepts* joined in this way must be from the same top level *hierarchy*. The syntax does not mandate which *concepts* in the *expression* should have associated *descriptions* and which should not so it is valid, but not advisable, to mix and match. For example, the following syntax is valid:

217724009 +297186008 | motorcycle accident |

The syntax allows spaces, tabs and carriage returns in most places. For example, the following examples have identical meaning to the one above:

217724009 + 297186008 | motorcycle accident |

217724009

+ 297186008

| motorcycle accident |

Using the "+" symbol is symmetrical and equivalent to starting with one of the *concepts* and adding an | is a | *refinement*, with a *value* set to the other *concept*. For example, the following two *expressions* are equivalent to each other and to the preceding *expression*:

217724009 | accident caused by blizzard |:

116680003 | is a | =297186008 | motorcycle accident |

297186008 | motorcycle accident |:

116680003 | is a | =217724009 | accident caused by blizzard |

One or more *refinements* may be added to a *concept* to qualify it. This is done by putting the *concept* to be qualified before a colon and the qualifying *expression* after. The qualifying *expression* is of the form "attribute = value". The example below describes an operation to remove an ovary using a laser.

83152002 | oophorectomy |:

260686004 | method |=257820006| laser excision - action |

Refinements may also be applied to a conjoined *concept*. For example, the following two *expressions* (building on a preceding example) are equivalent:

313056006 | epiphysis of ulna |:

272741003 | laterality | =7771000 | left |

119189000 | ulna part | + 312845000 | epiphysis of upper limb |:

272741003 | laterality | =7771000 | left |

Note that there are no brackets round "119189000 | ulna part | + 312845000|epiphysis of upper limb" in the above example.

Where more than one qualifying *expression* is required, these can be separated using a comma. The example below describes the removal of the right ovary using laser excision.

83152002 | oophorectomy |:

260686004 | method |=257820006| laser excision - action |,

363704007 | procedure site | =20837000 | structure of right ovary |

A further example, below, describes the removal of the left fallopian tube using diathermy excision:

120053002 | Salpingectomy |:

260686004 | method | =261519002 | diathermy excision - action |,

363704007 | procedure site | =113293009 | structure of left fallopian tube |

Where a *SNOMED CT concept* comprises a number of other *concepts* or sub - *expressions*, it may be necessary to group qualifications applied to that *concept* in order to avoid ambiguity as to how they apply. An example of a *SNOMED CT concept* that comprises a number of other sub - *expressions* is:

116028008 | salpingo-oophorectomy |

This procedure comprises two sub-procedures: the excision of part of all of the ovarian structure; and the excision of part or all of the fallopian tube structure. We should note at this point that there is a subtle difference between a *subsumptive relationship* and a *comprising relationship*:

A motorcycle accident caused by low visibility **is a** motorcycle accident AND

is an accident caused by a blizzard.

A salpingo-oophorectomy **comprises** a fallopian tube excision and an oophorectomy.

This is demonstrated by the *SNOMED CT normal form* for salpingo-oophorectomy, as shown below:

71388002 | procedure |:

{260686004 | method | =129304002 | excision - action |,

405813007 | procedure site - Direct | =15497006 | ovarian structure |}

{260686004 | method | =129304002 | excision - action |,

405813007 | procedure site - Direct | =31435000 | fallopian tube structure |}

Where it is necessary within a *postcoordinated expression* to unambiguously qualify individual components of a *concept* comprised of a number of other *expressions* (as in the above example), grouping may be used. The following example describes a salpingo-oophorectomy, with laser excision of right ovary and diathermy

excision of left fallopian tube. Note that without the grouping, it would not be possible to tell on what structure the laser excision was used and on what structure the diathermy excision was used.

116028008 | salpingo-oophorectomy |:

{260686004 | method |=257820006| laser excision - action |,

363704007 | procedure site |=20837000 | structure of right ovary |}

{260686004 | method |=261519002 | diathermy excision - action |,

363704007 | procedure site |=113293009 | structure of left fallopian tube |}

A number of grouped *qualifiers* may be thus used to refine a *concept*. Note there is no comma between adjacent groups (as there are between adjacent *expressions*). Also note, the syntax does not limit the number of *qualifiers* in a group or the number of groups within an *expression*.

It is also possible to nest *expressions*, one inside the other. Any legal *expression* may be wrapped in a pair of brackets, and included in another *expression* in the same way as a *concept* would be. For example, the following *expression* describes a fracture of the femur caused by a motorcycle accident in a blizzard:

71620000 | fracture of femur |:

42752001 | due to |= (217724009 | accident caused by blizzard | +297186008 | motorcycle accident |)

In the example above, note the use of "()" brackets, to identify a nested *expression*, as opposed to "{}" brackets, used elsewhere, to identify groups.

The following examples show how complex *expressions* may be build up from simple ones, a layer at a time. This first *expression* describes a left hip:

24136001 | hip joint structure |:

272741003 | laterality |=7771000 | left |

This next uses the "left hip" *expression* to describe a procedure to replace it:

397956004 | prosthetic arthroplasty of the hip |:

363704007 | procedure site |= (24136001 | hip joint structure | :272741003 | laterality |=7771000 | left |)

Applying a further grouped *refinement* to the above describes a procedure to replace a left hip by inserting a prosthesis. Note that this example mixes an ungrouped qualification and a grouped qualification. Where this is done, all ungrouped qualifications should appear before the groups. Note also that there is no comma between the last qualification and the first group.

397956004 | prosthetic arthroplasty of the hip |: 363704007 | procedure site |= (24136001 | hip joint structure | :272741003 | laterality |=7771000 | left |) {363699004 | direct device |=304120007 | total hip replacement prosthesis |,

260686004 | method |=257867005 | insertion - action |}

Finally, the above *expression* may be included within a contextual wrapper, to describe a procedure that has been performed on a patient to replace a left hip by inserting a prosthesis.

243796009 | situation with explicit context |: {363589002 | associated procedure |= (397956004 | prosthetic arthroplasty of the hip |: 363704007 | procedure site |= (24136001 | hip joint structure | :272741003 | laterality |=7771000 | left |) {363699004 | direct device |=304120007 | total hip replacement prosthesis |,

260686004 | method |=257867005 | insertion - action |}), 408730004 | procedure context |=385658003 | done |, 408731000 | temporal context |=410512000 | current or specified |, 408732007 | subject relationship context |=410604004 | subject of record | }

4.3.4.2 Expression in definition forms



An *expression* can be transformed to definition form and the representations applicable to this alternative form can then be applied. However, this approach is limited because several of the forms used to represent *concept* definitions do not support nesting.

4.3.4.3 Human-readable renderings



An *expression* may be rendered according to particular rules to generate human-readable representations.

Specific "simple" rules have been specified by NHS Connecting for Health in the UK. Alternative suggestions for more natural rendering have also been made to extend this initial outline proposal.

Advice on this topic may be added to future revisions of this guide.

4.3.5 Stated Relationships Guide



This part of the Guide provides information about the Stated *relationships tables* and the Web Ontology Language (OWL) transformation.

4.3.5.1 Stated Relationships File



The *Stated Relationship File* contains the *stated form* of SNOMED CT. The *stated form* of a *Concept* is the *Description Logic* definition that is directly edited by authors or editors. It consists of the stated | Is a | *relationships* plus the defining *relationships* that exist prior to running a *classifier* on the logic definitions. Therefore, the *stated form* of a *Concept* is represented by a collection of *relationships*: one or more | Is a | *relationships* and zero or more defining *relationships*.

The *Stated Relationship File* is in the same table format as the *Relationship File*, but the value of the *characteristicTypeld* field is | Stated relationship (core metadata concept) |.

The *stated form* enables implementers to test a *classifier* for consistency, by comparing the results of classification with the distributed *Relationship File*, which is the inferred form.

: Implementers should **not** use the *Stated Relationships File* unless they understand the implications of using this and provide software which makes *Description Logic* inferences from the *stated form*. The standard distribution form (the *Relationships File*) provides a *inferred view* which includes inferences derived from the *stated form*.

4.3.5.2 Description Logic (OWL or KRSS) Transform



The *Description Logic* Transform Script, written in the Perl language, performs a transform of the Stated *Relationships* into Web Ontology Language (OWL) format or KRSS format. There are two options for the syntax of the OWL output: RDF/XML, or OWL Functional Syntax. The RDF/XML is more verbose and results in a file approximately double the size of the Functional Syntax file.

4.3.5.2.1 Object Properties



SNOMED CT's attributes, the middle element of the *concept-Relationship-concept* triple, correspond to OWL Object Properties. The hierarchy under 410662002 | concept model attribute | contains all the attributes that have been approved for use as object properties. In addition, the *subtype Relationships* (i.e. | Is a | *Relationships*) between attributes in the | concept model attribute | hierarchy, as expressed in the stated *relationships tables*, are used by the script to automatically generate the corresponding sub-property axioms in OWL. For example, | Procedure site - Direct | appears as a *subtype* of | PROCEDURE SITE | in the stated *relationships tables*, and so the script automatically makes the OWL object property 'PROCEDURE SITE DIRECT' a sub-property of OWL object property 'PROCEDURE SITE'.

4.3.5.2.2 Relationship Grouping



When transforming *Relationships* to OWL or KRSS, all rows that have a *RelationshipType* that are allowed to be grouped, even if a particular row is ungrouped (i.e. even if the row has a *RelationshipGroup*

value meaning ‘ungrouped’), must be nested under an existential restriction that represents the (potential) grouping. This existential restriction is labeled with the OWL object property called ‘RoleGroup’. It is just another attribute, in the sense that it has a *SNOMED CT identifier* and is named in the distributed *concepts table*. In KRSS syntax, the stated definition of myConceptId1 with a stated definition that has a row with the triplet consisting of myConceptId1, myRelationshipType, myConceptId2 would translate into:

```
(defprimconcept myConceptId1
  (and parentConceptId
    (some RoleGroupId
      (some myRelationshipType myConceptId2))))
```

Attributes that are never grouped:

All RelationshipTypes are allowed to be grouped except | IS A |, and the following four:

- 123005000 | PART OF |
- 272741003 | LATERALITY |
- 411116001 | HAS DOSE FORM |
- 127489000 | HAS ACTIVE INGREDIENT |

4.3.5.2.3 Right Identities



There has historically been limited use of right identities, also known as property chains, in *SNOMED CT*. The one property chain that is in the current release is | DIRECT SUBSTANCE | o | HAS ACTIVE INGREDIENT | -> | DIRECT SUBSTANCE |. The OWL transform properly represents this property chain in the OWL 2 EL Profile. It is not yet represented in the *relationships tables*, or anywhere else in the RF1 or RF2 format *SNOMED CT distribution files*. This is a recognized deficiency which has not yet been addressed partly because there is only one such declaration, and no inferences in standard release are affected by this single right identity declaration.

4.3.5.2.4 Running the Perl transform script



Run the script according to the pattern:

```
perl <scriptfilename> <arg0> <arg1> <arg2> <arg3> <arg4>
```

where

- <scriptfilename> is the name of the file containing the transform script
 - In the July 2013 release this Perl script file is named:
tls2_StatedRelationshipsToOwlKRSS_INT_20130731.pl
- <arg0> can be KRSS, OWL, or OWLF
 - KRSS causes output to be formatted according to KRSS2 which is parsable by the OWL API 3.2.2, or by CEL or other classifiers
 - OWL causes output to be formatted according to OWL XML/RDF
 - OWLF causes output to be formatted according to the OWL functional syntax
- <arg1> is the name of the file containing the RF2 format *SNOMED CT Concepts Table* snapshot
 - In the July 2013 release this file is named: sct2_Concept_Snapshot_INT_20130731.txt
- <arg2> is the name of the file containing the RF2 format *SNOMED CT Descriptions Table* snapshot
 - In the July 2013 release this file is named: sct2_Description_Snapshot-en_INT_20130731.txt
- <arg3> is the name of the file containing the RF2 format *SNOMED CT Stated relationships tables*
 - In the July 2013 release this file is named: sct2_StatedRelationship_Snapshot_INT_20130731.txt
- <arg4> is the name of the output file. Any valid file name can be used.
 - for example: res_StatedOWLF_INT_20130731.owl

An example execution command on a Windows machine, to produce the *stated view* of *SNOMED CT* according to OWL Functional syntax, would then look like the following:

```
C:\> perl tIs2_StatedRelationshipsToOwlKRSS_INT_20130731.pl OWLF
    sct2_Concept_Snapshot_INT_20130731.txt
    sct2_Description_Snapshot-en_INT_20130731.txt
    sct2_StatedRelationship_Snapshot_INT_20130731.txt
    res_StatedOWLF_INT_20130731.owl
```

4.3.5.2.5 Importing into an editor



Once the output file has been successfully created (e.g. *res_StatedOWLF_INT_20130731.owl*), an ontology editor that uses the OWL API should be able to import the file, assuming that the editor can handle very large files and that it is configured to use large amounts of memory, and your system has adequate memory (see FAQ below). The current version of the transform script has been tested with Protege running the OWL API version 3.2.2 and the OWL 2 Profile is OWL 2 EL. The table below presents the metrics that result from the July 2013 beta release. The final release may have a few minor differences.

Table 22: Metrics to Validate Import of SNOMED OWL, July 2013 International Release (20130731) - beta

Protege Ontology Metrics	Value
Class count	297327
Object property count	62
DL expressivity	ALER
SubClass axioms count	229247
Equivalent class axioms count	68068
Sub object property axioms count	11
Object property chain subproperty axioms count	1
Annotation Assertion axioms count	297378

4.3.5.2.6 SNOMED CT OWL Distribution FAQ



4.3.5.2.6.1 Access



1. Where do I obtain a copy of the OWL version of *SNOMED CT*?

- You can currently **generate** an OWL version of *SNOMED CT* using the Perl script and '*stated view*' file in the standard distribution of *SNOMED CT*.
- The Perl script and *Stated Relationships File* are distributed in the main release in different directories. The script is located in a folder called 'Resources/StatedRelationshipsToOwlKRSS/' and the RF2 snapshot files for *concepts*, *Descriptions* and stated *Relationships* are located in a folder called 'RF2Release/Snapshot/Terminology'. Prior to the January 2012 release, the transform was based on an RF1 format stated *Relationships* file - see documentation of prior releases for historical data and transform scripts.

2. What do you mean I need to 'generate' the OWL version of *SNOMED CT*?

- The OWL version of *SNOMED CT* is currently not distributed with the core release. However you can generate a local OWL version of *SNOMED CT* by executing the Perl script mentioned above. The instructions for using the Perl script are included in the *Stated Relationships Guide* (part of the Technical Implementation Guide), and also as comments in the header of the file containing the Perl script, which can be viewed in your favorite text editor (e.g. Notepad, Wordpad, etc).

3. What do I need to generate the OWL version of *SNOMED CT*?

- In order to generate the OWL version of *SNOMED CT*, you will need to have ‘Perl’ installed on your machine.
 - In addition to the *Stated Relationships file* and Perl script mentioned, you will also require the RF2 (*Release Format 2*) version of the *concepts table* and *descriptions table*, named ‘sct2_Concept_Snapshot_INT_yyyyymmdd.txt’, and ‘sct2_Description_Snapshot-en_INT_yyyyymmdd.txt’ in the January 2012 *International Release* and subsequently. These are found in the ‘RF2Release/Snapshot/Terminology’ folder of the *International Release*.
- 4. I get errors when I try to generate the OWL version using the Perl script. What can I do?**
- Please check the following, before you report errors in the build process:
 - Ensure you have Perl properly installed on your machine.
 - Ensure that you are using versions of the Perl script, *Stated Relationships*, *Concepts*, and *Descriptions* all from the same release date and same *Release Format* (i.e. RF2). You will definitely get errors if you try to use a script designed for RF2 on RF1 format files, and vice versa. There is no guarantee of backwards compatibility of the script - i.e. a version released for use with July 2013 RF2 files may not work with prior release RF2 files.
 - Errors may be reported on the *IHTSDO Collaborative Space*, under the Implementation SIG site (in the General Discussions Forum).

4.3.5.2.6.2 Licensing



1. What are the license restrictions on the OWL version of *SNOMED CT*?

- There is a single world-wide license for *SNOMED CT* for all purposes, called the “affiliate license”. The same license applies to the OWL version of *SNOMED CT*. You can find it by following the highlighted link labelled “*SNOMED CT Affiliate License Agreement*” on the right hand side of the page at www.ihtsdo.org/join-us/use-snomed-ct-licenses

4.3.5.2.6.3 Importing and Visualization



1. How do I load and visualize *SNOMED CT* in OWL format?

- Though the KRSS or OWL files generated by the perl script can be viewed in a text editor, in order to sensibly visualize the OWL release you require a tool like Protégé 4(<http://protege.stanford.edu/>). Please note that version 4 of Protégé is required to load and visualize *SNOMED CT*.

2. Protégé crashes (or becomes unresponsive) when I try to visualize the class hierarchy on my machine!

- Protégé is known to take some time to generate the class hierarchy for display. It might be worthwhile increasing the memory allocation of Protégé before your start loading *SNOMED CT*. Please refer to the relevant Protégé documentation for exact details on increasing maximum memory available to Protégé.

3. Help, the hierarchies are rendered as *concept IDs* in Protégé! How can I change this into *fully specified names*?

- You need change the rendering options in Protégé to render using ‘labels’. In order to do that in Protégé 4, select ‘Render using annotation values’ from ‘Preferences/Renderer/Entity rendering’.

4.3.5.2.6.4 Classification



1. What DL reasoners are currently supported for classifying OWL version of *SNOMED CT*?

- There are Protégé 4.1 plugins for several DL reasoners that can classify *SNOMED CT* provided the machine specifications listed below are met. These include Snorocket, ELK, and Fact++.

2. How long does it take to classify *SNOMED CT* in Protégé 4.1?

- That depends on the classifier and how fast your machine is. Both Snorocket and ELK are very fast, and complete in well under 30 seconds (actual clock time) on an adequately configured machine.

3. How do I use the DL Query Tab in Protégé 4 to create *postcoordinated expressions*?

- a. We recommend looking at the Protege OWL Tutorial (<http://www.co-ode.org/resources/tutorials/ProtegeOWLTutorial.pdf>) for more information on using Protege 4.1 to construct *expressions*. In the Protege world, *postcoordinated expressions* are referred to as *DL expressions*.
- b. In order to create *postcoordinated expressions* in the DL query tab, you are required to use the Manchester syntax for the *expressions*. In order to understand the Manchester syntax, you will need to read and work the examples in the Protege OWL tutorial.

4. What can I do once I have classified *SNOMED CT* in Protégé 4.1?

- That depends on what you intended to do with a classified version of *SNOMED CT*. Within Protege 4.1, you can do subsumption testing over arbitrary DL *expressions* using the 'DL query tab' among other things. This feature might be used to implement subsumption testing over *postcoordinated expressions*.

4.3.5.2.6.5 Machine specification



1. What are the minimum specifications of machines for viewing loading and viewing *SNOMED CT* in OWL?

- As a general rule, for reasonable performance, one would require a 64-bit machine, such as an Intel Core 2 Duo, with clock speed of 2GHz or more and 4GB of RAM to load the OWL version of *SNOMED CT* in Protégé.
- The actual memory requirements might actually be smaller depending on your machine. Users have successfully loaded *SNOMED CT* on a 32-bit Mac OS X machine with 2GB RAM, and on a 32-bit Linux (Ubuntu) machine with 3GB RAM. However, display and editing performance is usually considered unacceptably slow when using these minimal configurations.
- Loading and visualizing the OWL version of *SNOMED CT* using alternate methods might have different machine specifications.

2. What are the minimum specifications for classifying *SNOMED CT*?

- It is believed that one would require a 64-bit machine with an Intel Core 2 Duo processor (or better) with 4GB of RAM to classify *SNOMED CT* using the classifiers bundled with Protégé 4. Users have successfully classified *SNOMED CT* on a 32-bit Mac OS X machine with 2GB RAM, and on a 32-bit Linux (Ubuntu) machine with 3GB of RAM.

4.3.5.2.6.6 Software



1. Can I bundle the OWL version of *SNOMED CT* in my open source software?

- *SNOMED CT* is licensed under the affiliate license described above. *SNOMED CT* or any derivatives of *SNOMED CT* cannot be redistributed under any other license (including any form of open source license).

2. Am I allowed to make extensions or modification to the OWL release of *SNOMED CT* and include it in my software?

- *SNOMED CT* is licensed under the affiliate license described above. *SNOMED CT* or any derivatives of *SNOMED CT* cannot be redistributed under any other license (including any form of open source license).

3. What API can I use to programmatically access the OWL version of *SNOMED CT*?

- Though there are many candidate APIs available, most DL reasoners bundled with Protégé 4.1 use the Manchester OWL API (owlapi.sourceforge.net). There are examples online on how to load an ontology. It might also be possible to use the Jena API (jena.sourceforge.net) to load the RDF/XML version of the file.

4.3.6 Other Representational Forms



This section summarizes some of the other forms in which *SNOMED CT components* and *expressions* may be represented. This includes some references to a selection of proprietary and standard representation which have been used or suggested for particular uses. Mention in this section is intended to be illustrative and does not represent endorsement. Additional suggestions that may be helpful to some implementers could be added in future.

4.3.6.1 Complete Concept Representations



Representation of the *concept* as a whole includes the definition of the *concept* but also includes additional properties of *concepts* and associated components such as *descriptions* and *Reference Sets*.

As a rule representations of complete *SNOMED CT concepts* will be specific to *SNOMED CT*. Some of these representations will be specified by *SNOMED* and others will be application specific designs building on the *SNOMED CT* specifications. If generic forms of representation are used then guidelines on how particular properties from *SNOMED* are represented are necessary.

4.3.6.1.1 SNOMED CT distribution files



The *Release File Specifications* (5) provide a form of representation for complete *concepts* (and other *components*).

The *release files* are designed efficient for large scale batch distribution and facilitate easy import into relational databases. They may need to be indexed and optimized to provide a practical implementable representation.

4.3.6.1.2 IHTSDO workbench internal format



The set of database table used by the *IHTSDO Workbench* to maintain *SNOMED CT* include a full representation of all types of *SNOMED CT Components*. The representation is closely aligned with *SNOMED CT Release Format 2*. However, additional data is stored to manage workflow and conflict resolution during the development process.

4.3.6.1.3 SNOMED CT Distribution XML



The XML distribution schema specified by *SNOMED* provides a form of representation for complete *concepts* (including associated components).

The XML distribution files can be used as an alternative to the *SNOMED CT distribution files*. However, they are particularly efficient for communication of individual *concepts* or sets of concept (e.g. for update change-sets).

4.3.6.1.4 Application internal



SNOMED CT enabled applications will usually have their own internal optimized representation of the *SNOMED* distribution information. This may simply be a relational database with a specified set of indices or it may be a significantly different form.

Examples of proprietary representation include the forms used internally by CliniClue (ClueData), Health Language, Apelon TDE and other implementations.

4.3.6.1.5 Various human-readable renderings



Concept information may be rendered in various ways to allow human visualization and understanding. These forms may include plain text, mark-up and graphical trees diagramming *relationships*. All of these renderings can be regarded as representations of complete *concepts* or their definitions.

4.3.6.2 Concept Definition Representations

See also: [Complete concept representations](#)



4.3.6.2.1 KRSS

KRSS is a general form for representing logical *descriptions*.

Transforms have been developed internally for producing KRSS representations of *SNOMED CT* definitions (see [Stated Relationships Guide](#)).

4.3.6.2.2 OWL

The Web Ontology Language (OWL) is a web-technology based approach to representation of logical *concept definitions*.

Transforms have been developed internally for producing OWL representations of *SNOMED CT* definitions (see [Stated Relationships Guide](#)).

4.3.6.2.3 Representing Definitions as Expressions

A *Concept definition* can also be represented as an *expression* (see [Representational Forms for Expressions](#)). One or more of the *supertype parent concepts* are represented as *focus concepts* and other defining *relationships* are represented as refining *attributes*.



4.3.6.2.4 Various human-readable renderings

Concept definitions may be rendered in various ways to allow human visualization and understanding. These forms may include plain text, mark-up and graphical trees diagramming *relationships*. All of these renderings can be regarded as representations of *concept definitions*.



4.3.7 Additional Reference Materials



4.3.7.1 Introduction

This section contains additional technical information that does not referenced by other part of this guide.



4.3.7.2 Unicode UTF-8 encoding

4.3.7.2.1 Introduction



UTF-8 is an efficient encoding of *Unicode* character - *strings* that recognizes the fact that the majority of text-based communications are in ASCII. It therefore optimizes the encoding of these characters. *UTF-8* is supported by many 32-bit Windows® applications.

Unicode is preferred to ASCII because it permits the inclusion of accents, scientific symbols and characters used in *languages* other than English. The *UTF-8* format is a standard encoding that provides the most efficient means of encoding 16-bit *Unicode* characters in cases where the majority of characters are in the ASCII range. Both *UTF-8* and the alternative *UTF-16* encoding are supported by modern 32-bit operating systems such as Windows® 95, 98 and NT.

SNOMED CT uses *UTF-8* characters.

4.3.7.2.2 Character encoding



ASCII characters are encoded as a single byte.

- Greek, Hebrew, Arabic and most accented European characters are encoded as two bytes;
- All other characters are encoded as three bytes;
- The individual characters are encoded according to the following rules.

4.3.7.2.2.1 Single byte encoding



Characters in the *range* 'u+0000' to 'u+007f' are encoded as a single byte.

Table 23: UTF-8 Single Byte Encoding

byte 0	
0	bits 0-6

4.3.7.2.2.2 Two byte encoding

Characters in the range 'u+0080' to 'u+07ff' are encoded as two bytes.

Table 24: Two byte encoding

byte 0			byte 1		
1	1	0	bits 6-10		

4.3.7.2.2.3 Three byte encoding

Characters in the range 'u+0800' to 'u+ffff' are encoded as three bytes:

Table 25: UTF-8 Three Byte Encoding

byte 0				byte 1			byte 2		
1	1	1	0	bits 12-15				1	0

4.3.7.2.3 Notes on encoding rules

The first bits of each byte indicate the role of the byte. A zero bit terminates this role information.

Thus possible byte values are:

Table 26: UTF-8 Encoding Rules

Bits	Byte value	Role
0???? ?? ?	000-127	Single byte encoding of a character
10??? ?? ?	128-191	Continuation of a multi-byte encoding
110?? ?? ?	192-223	First byte of a two byte character encoding
1110? ?? ?	224-239	First byte of a three byte character encoding
1111? ?? ?	240-255	Invalid in UTF-8

4.3.7.2.4 Example encoding



Table 27: UTF-8 Encoding Example

Character	S	C	T	®				
Unicode	0053	0043	0054	00AE		2462		
Bytes	01010011	01000011	01010100	11000010	10101110	11101111	10111111	10111111

4.3.7.3 Check-digit Computation



The *SCTID* (See [Component features - Identifiers](#) on page 48) includes a *check-digit*, which is generated using Verhoeff's dihedral check. This section explains the algorithm used and includes sample source code for generating and checking the *check-digit* in Java Script and Microsoft Visual Basic.

4.3.7.3.1 Verhoeff's Dihedral Group D5 Check



The mathematical *description* of this technique may appear complex but in practice it can be reduced to a pair of two-dimensional arrays, a single dimensional inverse array and a simple computational procedure. These three arrays are shown in the following tables.

- The first array contains the result of “Dihedral D5” multiplication;
- The second array consists of 8 rows of which two are defined while the rest are derived by applying the following formula: $F(i, j) = F(i - 1, F(1, j))$;
- The third array consists of a single row containing the inverse of the Dihedral D5 array it identifies the location of all the zero values in the first array.

Table 28: Results of Dihedral D5 multiplication

	0	1	2	3	4	5	6	7	8	9
0	0	1	2	3	4	5	6	7	8	9
1	1	2	3	4	0	6	7	8	9	5
2	2	3	4	0	1	7	8	9	5	6
3	3	4	0	1	2	8	9	5	6	7
4	4	0	1	2	3	9	5	6	7	8
5	5	9	8	7	6	0	4	3	2	1
6	6	5	9	8	7	1	0	4	3	2
7	7	6	5	9	8	2	1	0	4	3
8	8	7	6	5	9	3	2	1	0	4
9	9	8	7	6	5	4	3	2	1	0

Table 29: The full array for Function F

	0	1	2	3	4	5	6	7	8	9
0	0	1	2	3	4	5	6	7	8	9
1	1	5	7	6	2	8	3	0	9	4
2	5	8	0	3	7	9	6	1	4	2
3	8	9	1	6	0	4	3	5	2	7
4	9	4	5	3	1	2	6	8	7	0
5	4	2	8	6	5	7	3	9	0	1
6	2	7	9	3	8	0	6	4	1	5
7	7	0	4	6	9	1	3	2	5	8

Table 30: The Inverse D5 array

0	1	2	3	4	5	6	7	8	9
0	4	3	2	1	5	6	7	8	9

The *Identifier* is checked by starting at the rightmost digit of the *Identifier* (the *check-digit* itself) and proceeding to the left processing each digit as follows:

- *Check* = `ArrayDihedralD5 (Check, ArrayFunctionF((Position Modulus 8), Digit))`
Check = the running value of the check-sum (starts at zero and modified by each step).
Position = the position of the digit (counted from the right starting at zero).
Digit = the value of the digit.

The final value of *Check* should be zero. Otherwise the check has failed.

When calculating the *check-digit* the same process is applied with a minor variation:

- *Position* is the position that the digit will have when the *check-digit* has been appended.
- The final value of *Check* is applied to the Inverse D5 array to find the correct *check-digit*.
Check-digit = `ArrayInverseD5 (Check)`.

4.3.7.3.2 Sample Java Script for computing Verhoeff's Dihedral Check



The script is presented here as part of an HTML page.

 **Note:**

The code below can be used by copying all the lines in the above section into an HTML file and opening this with a web browser. From the HTML version of this guide the following link provides access to this file as an [web page](#).

```
<!DOCTYPE html SYSTEM "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html>
  <head>
    <title>SNOMED CT Identifier Check</title>
    <style>
      body{font-family:Arial, Helvetica, sans-serif}
    }
    </style>
    <meta content="text/html; charset=iso-8859-1" http-equiv="Content-Type"><meta>
      <script type="text/javascript" language="JavaScript">

        var FnF = new Array();
        FnF[0] = [0, 1, 2, 3, 4, 5, 6, 7, 8, 9];
        FnF[1] = [1, 5, 7, 6, 2, 8, 3, 0, 9, 4];
        for ( var i = 2; i < 8; i++ )
        {
          FnF[i] = [,,,,,,,,,];
          for ( var j = 0; j < 10; j++ )
            FnF[i][j] = FnF[i - 1][FnF[1][j]];
        }
        var Dihedral = new Array(
          [0, 1, 2, 3, 4, 5, 6, 7, 8, 9],
          [1, 2, 3, 4, 0, 6, 7, 8, 9, 5],
          [2, 3, 4, 0, 1, 7, 8, 9, 5, 6],
          [3, 4, 0, 1, 2, 8, 9, 5, 6, 7],
          [4, 0, 1, 2, 3, 9, 5, 6, 7, 8],
          [5, 9, 8, 7, 6, 0, 4, 3, 2, 1],
          [6, 5, 9, 8, 7, 1, 0, 4, 3, 2],
          [7, 6, 5, 9, 8, 2, 1, 0, 4, 3],
          [8, 7, 6, 5, 9, 3, 2, 1, 0, 4],
          [9, 8, 7, 6, 5, 4, 3, 2, 1, 0] );
        var InverseD5 = new Array(0, 4, 3, 2, 1, 5, 6, 7, 8, 9 );

        function VerhoeffCheck()

        {
          var check = 0;
          var IdValue = document.form.numcd.value;
          document.getElementById("out").innerText = "";
          document.getElementById("out").setAttribute("style", "color :red;");
          document.getElementById("component").innerText = "Invalid partition";
          document.getElementById("component").setAttribute("style", "color :green;");
          document.getElementById("extnamespace").innerText = "No namespace";
          document.getElementById("extnamespace").setAttribute("style", "color :red;");

          for ( var i=IdValue.length-1; i >=0; i-- )
            check = Dihedral[check][FnF[(IdValue.length-i-1) % 8][IdValue.charAt(i)]];
          if ( check != 0 ) { document.getElementById("out").innerText = "Check-digit ERROR"; }
          else if ( IdValue.length < 6 ) {document.getElementById("out").innerText = "SCTID too short";}
          else if ( IdValue.length > 18 ) {document.getElementById("out").innerText = "SCTID too long";}
          else {document.getElementById("out").innerText = "Check-digit OK";
            document.getElementById("out").setAttribute("style", "color :green;");
            switch (IdValue.substr(IdValue.length-3,2))
            {

```

```

case "00":
    document.getElementById("component").innerText ="Concept";
    document.getElementById("exnamespace").innerText ="International";
    break;
case "01":
    document.getElementById("component").innerText ="Description";
    document.getElementById("exnamespace").innerText ="International";
    break;
case "02":
    document.getElementById("component").innerText ="Relationship";
    document.getElementById("exnamespace").innerText ="International";
    break;
case "03":
    document.getElementById("component").innerText ="Subset (RF1)";
    document.getElementById("exnamespace").innerText ="International";
    break;
case "04":
    document.getElementById("component").innerText ="Cross~Map Set (RF1)";
    document.getElementById("exnamespace").innerText ="International";
    break;
case "05":
    document.getElementById("component").innerText ="Cross~Map Target (RF1)";
    document.getElementById("exnamespace").innerText ="International";
    break;
case "10":
    document.getElementById("component").innerText ="Concept";
    document.getElementById("exnamespace").innerText =IdValue.substr(IdValue.length-10,7);
    break;
case "11":
    document.getElementById("component").innerText ="Description";
    document.getElementById("exnamespace").innerText =IdValue.substr(IdValue.length-10,7);
    break;
case "12":
    document.getElementById("component").innerText ="Relationship";
    document.getElementById("exnamespace").innerText =IdValue.substr(IdValue.length-10,7);
    break;
case "13":
    document.getElementById("component").innerText ="Subset (RF1)";
    document.getElementById("exnamespace").innerText =IdValue.substr(IdValue.length-10,7);
    break;
case "14":
    document.getElementById("component").innerText ="Cross~Map Set (RF1)";
    document.getElementById("exnamespace").innerText =IdValue.substr(IdValue.length-10,7);
    break;
case "15":
    document.getElementById("component").innerText ="Cross~Map Target (RF1)";
    document.getElementById("exnamespace").innerText =IdValue.substr(IdValue.length-10,7);
    break;
default:
    document.getElementById("component").setAttribute("style", "color :red;");
}
if (document.getElementById("exnamespace").innerText=='International')
{document.getElementById("exnamespace").setAttribute("style", "color :green;");}
else if (IdValue.length>10) {document.getElementById("exnamespace").setAttribute("style", "color :green;");}

else {document.getElementById("exnamespace").innerText="Invalid Namespace";
}
}
}

function VerhoeffCompute( )
{
    var IdValue = document.form.num.value; var check = 0;
    document.form.numcd.value= "";
}

```

```

for ( var i = IdValue.length-1; i >=0; i-- )
    check = Dihedral[check][FnF[(IdValue.length-i) % 8][IdValue.charAt(i)]];
    document.form.numcd.value = document.form.num.value + InverseD5[check];
    VerhoeffCheck();
    document.getElementById("out").innerText = "Computed check-digit";
}
</script>
</head>
<body>
    <h1>SNOMED CT Identifier Check</h1>
    <form action="" name="form">
        <table border="1" width="441">
            <tr>
                <td width="212" height="25">
                    Partial Identifier <br/>(without check-digit)&nbsp;
                </td>
                <td width="115" height="25">
                    <input name="num" size="18"/>
                </td>
                <td width="92" height="25">
                    <input onclick= "VerhoeffCompute()" type="button" value="Compute"/>
                </td>
            </tr>
            <tr>
                <td width="212" height="35">
                    SNOMED CT Identifier
                </td>
                <td width="115" height="35">
                    <input name="numcd" size="18"/>
                </td>
                <td width="92" height="35">
                    <input onclick= "VerhoeffCheck()" type="button" value="Check"/>
                </td>
            </tr>
            <tr>
                <td width="212" height="23">
                    Result of check&nbsp;
                </td>
                <td width="115" height="23" colspan="2" id="out">
                </td>
            </tr>
            <tr>
                <td width="212" height="23">
                    Component type
                </td>
                <td width="115" height="23" colspan="2" id="component">
                </td>
            </tr>
            <tr>
                <td width="212" height="23">
                    Namespace
                </td>
                <td width="115" height="23" colspan="2" id="exnamespace">
                </td>
            </tr>
        </table>
        <p style="margin-left: 0; margin-right: 0">
            This Verhoeff checking part of this code was based on a webpage at:
        </p>
        <ul>
            <li>
                <a href="http://www.augustana.ab.ca/~mohrj/algorithms/checkdigit.html">
                    http://www.augustana.ab.ca/~mohrj/algorithms/checkdigit.html
                </a>
            </li>
        </ul>
    </form>
</body>

```

```

        </ul>
    </form>
</body>
</html>

```

4.3.7.3.3 Sample Visual Basic for computing Verhoeff's Dihedral Check



Option Explicit

Private Dihedral(9) As Variant

Private FnF(7) As Variant

Private InverseD5 As Variant

Public Function VerhoeffCheck(ByVal IdValue As String) As Boolean

'Check the supplied value and return true or false

Dim tCheck As Integer, i As Integer

VerhoeffArrayInit

For i = Len(IdValue) To 1 Step -1

tCheck = Dihedral(tCheck)(FnF((Len(IdValue) - i) Mod 8)(Val(Mid(IdValue, i, 1))))

Next

VerhoeffCheck = tCheck = 0

End Function

Public Function VerhoeffCompute(ByVal IdValue As String) As String

'Compute the check digit and return the identifier complete with check-digit

Dim tCheck As Integer, i As Integer

VerhoeffArrayInit

For i = Len(IdValue) To 1 Step -1

tCheck = Dihedral(tCheck)(FnF((Len(IdValue) - i + 1) Mod 8)(Val(Mid(IdValue, i, 1))))

Next

VerhoeffCompute = IdValue & InverseD5(tCheck)

End Function

Private Sub VerhoeffArrayInit()

'Create the arrays required

Dim i As Integer, j As Integer

'if already created exit here

If VarType(InverseD5) >= vbArray Then Exit Sub

'create the DihedralD5 array

```

Dihedral(0) = Array(0, 1, 2, 3, 4, 5, 6, 7, 8, 9)
Dihederal(1) = Array(1, 2, 3, 4, 0, 6, 7, 8, 9, 5)
Dihederal(2) = Array(2, 3, 4, 0, 1, 7, 8, 9, 5, 6)
Dihederal(3) = Array(3, 4, 0, 1, 2, 8, 9, 5, 6, 7)
Dihederal(4) = Array(4, 0, 1, 2, 3, 9, 5, 6, 7, 8)
Dihederal(5) = Array(5, 9, 8, 7, 6, 0, 4, 3, 2, 1)
Dihederal(6) = Array(6, 5, 9, 8, 7, 1, 0, 4, 3, 2)
Dihederal(7) = Array(7, 6, 5, 9, 8, 2, 1, 0, 4, 3)
Dihederal(8) = Array(8, 7, 6, 5, 9, 3, 2, 1, 0, 4)
Dihederal(9) = Array(9, 8, 7, 6, 5, 4, 3, 2, 1, 0)

```

'create the FunctionF array

FNF(0) = Array(0, 1, 2, 3, 4, 5, 6, 7, 8, 9)

FNF(1) = Array(1, 5, 7, 6, 2, 8, 3, 0, 9, 4)

'compute the rest of the FunctionF array

```

For i = 2 To 7
    FnF(i) = Array(0, 0, 0, 0, 0, 0, 0, 0, 0, 0)
    For j = 0 To 9
        FnF(i)(j) = FnF(i - 1)(FnF(1)(j))
    Next
Next

'Create the InverseD5 array
InverseD5 = Array("0", "4", "3", "2", "1", "5", "6", "7", "8", "9")

End Sub

```



4.3.7.3.4 Reasons for using a check-digit

Although a user should rarely type the *SCTID*, experience suggests that from time to time this will happen. A user may also copy and paste an *SCTID*. There is a significant risk of errors in these processes and inclusion of a *check-digit* is intended to reduce the risk of such errors passing undetected. The choice of *check-digit* algorithm has been made to maximize the detection of common typographical errors. These have been analyzed by in a paper by J. Verhoeff ("Error Detecting Decimal Codes", *Mathematical Center Tract 29*, The Mathematical Center , Amsterdam, 1969) and subsequently cited in Wagner and Putter, ("Error Detecting Decimal Digits", CACM, Vol 32, No. 1, January 1989). These papers give a detailed categorization of the sorts of errors humans make in dealing with decimal numbers, based on a study of 12000 errors:

- single errors: a becomes b (60% to 95% of all errors).
- omitting or adding a digit (10% to 20%).
- adjacent transpositions: ab becomes ba (10% to 20%).
- twin errors: aa becomes bb (0.5% to 1.5%).
- jump transpositions: acb becomes bca (0.5% to 1.5%).
- jump twin errors: aca becomes bcb (below 1%).
- phonetic errors: a0 becomes 1a -similar pronunciation e.g. thirty or thirteen (0.5% to 1.5%).

In the explanations above, a is not equal to b, but c can be any decimal digit.

4.3.7.3.4.1 A brief comparison of check-digit effectiveness



4.3.7.3.4.1.1 The IBM Check



The check-sums used for credit cards (the IBM check) picks up the most common errors but miss some adjacent transpositions and many jump transpositions. Assuming the pattern of errors described above, on average it will miss between 4% and 5% of expected errors.

4.3.7.3.4.1.2 The ISBN Check (Modulus 11)



The ISBN modulus 11 (used for *UK NHS number*) picks up more errors than the IBM checksum. Leaving 2% to 3% of errors undetected. However, it generates a check-sum value of 0 to 10 and thus cannot be represented as a single *check-digit* in about 9% of cases. The ISBN convention is to use "X" to represent the *check-digit* value 10 but this is incompatible with an *integer* representation. The *UK NHS number* uses this check-sum but regards and number generating a check-sum of 10 as an invalid *Identifier*. This approach could be applied to the *SCTID* but this would render 9% of possible values unusable in each partition and *namespace*. This would prevent a simple sequence of values from being allocated as the "item *Identifier*" within each *namespace*. More significantly the unusable item *Identifiers* would differ in each *namespace* or partition and this would prevent simple transpositions of item *Identifiers* between partitions and *namespaces*. Partitions could be a useful way of distinguishing developmental and released components and revising the partition and recalculating the *check-digit* would then be an elegant way to activate these components for a distribution version. It seems unwise to prevent future development and maintenance by using a check-sum that will prevent this.

4.3.7.3.4.1.3 Verhoeff's Check



Verhoeff's check catches all single errors, all adjacent transpositions, over 95% of twin errors, over 94% of jump transpositions and jump twin errors, and most phonetic errors. Therefore, like modulus 11,

the Verhoeff check reduces the undetected error rate to 2% or 3%. Unlike modulus 11, it does this using a single decimal *check-digit* and without limiting the range of valid numbers.

The majority of the undetected errors with both modulus 11 and Verhoeff result from additions or omissions of digits. Any *check-digit* methods is likely to miss 10% of such errors and since these comprise 10% to 20%. The Verhoeff scheme also misses four jump twin errors involving digits with a difference of 5 (i.e. 050 vs. 505, 161 vs. 616, 272 vs. 727, and 494 vs. 949).

4.3.7.4 Search Support Tables

4.3.7.4.1 Overview



Effective implementation of *SNOMED CT* depends on the ease and speed with which users can locate the *terms* and *Concepts* that they wish to use. An essential contribution to meeting this requirement is the ability to perform rapid and flexible text searches.

A set of word search tables (indexes) are included in the Developer Toolkit. These tables are designed to facilitate development of effective search facilities while reducing duplication of effort. However, neither these tables, nor indices derived from them, are sufficient to meet the full range of search requirements. Meeting the needs of different users for appropriate methods for locating particular *Concepts* is an area in which competitive development is expected and welcomed. Developers may choose to use some or all of the word search tables distributed with *SNOMED CT* or may develop their own solutions independent of these tables.

The intention of the word search tables is to identify candidate matches among the *Descriptions* (or *Concepts*) of *SNOMED CT*. An application or coding engine will apply further filtering to these candidate matches to identify the matches to be selected or displayed. A balance must be made between specificity and completeness of a search. The *keyword* algorithm is intended to maximize the likelihood that the required *Concept* will be included in the candidate matches rather than to achieve precision.

Applications may filter candidate matches using techniques that are many and varied. Some may take account of non-textual characteristics (e.g. *Reference Sets*, *subtype Relationships* or *Relationships*) while others use more complex textual techniques (e.g. word order dependence, case dependence, complete phrase matching, regular-expression matching, Soundex). These extended text search techniques are beyond the scope of the *keyword* generation algorithm.

The algorithm for *keyword* generation is only applicable for English and other western European *languages*. It is not intended to apply to Russian, Greek, Slavic or to any non-European *languages*.

Please refer to the *Technical Implementation Guide* for additional search implementation guidance.

4.3.7.4.2 Search index - structure diagram

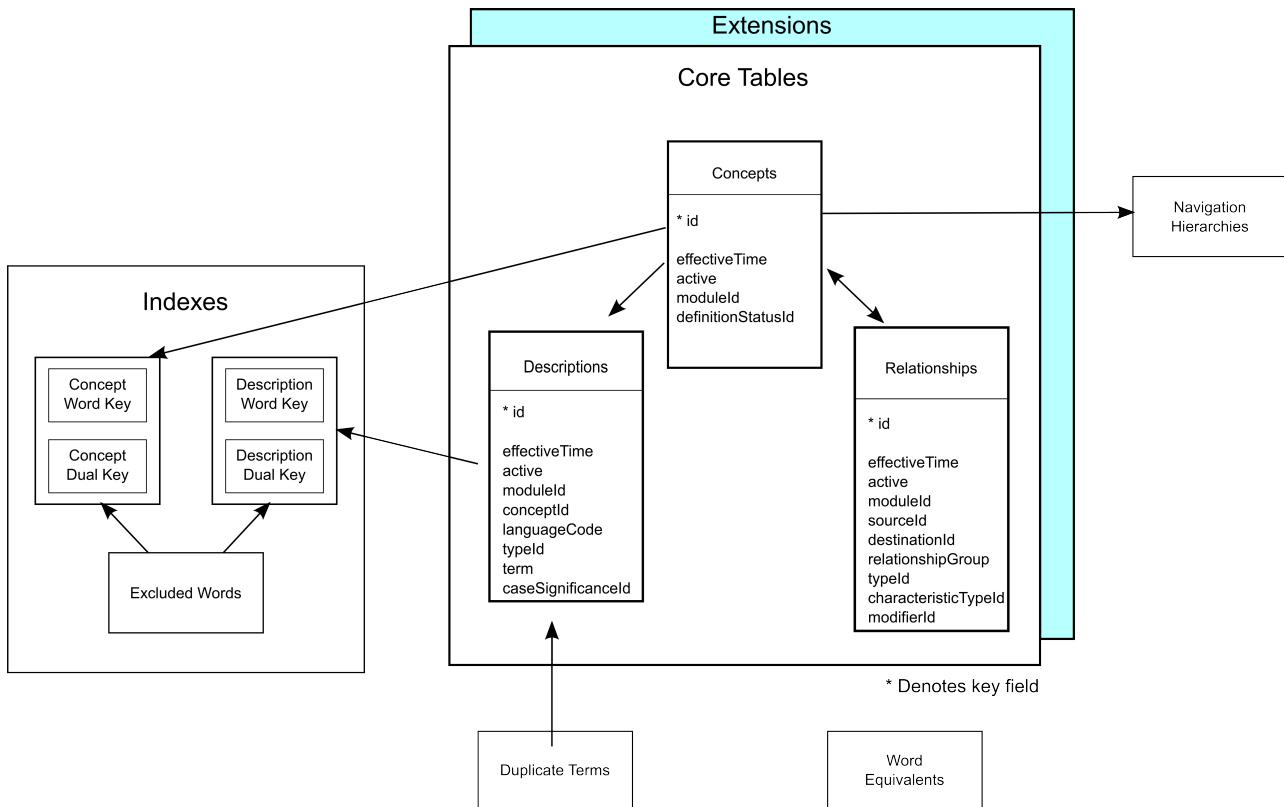


Figure 35: Search Index Overview (RF2)

4.3.7.4.3 Word Search Tables - Summary



The following five tables are included in the *Developer Toolkit* of *SNOMED CT*. These tables are derived from the *SNOMED CT Descriptions Table*. The *LanguageCode* of the *Descriptions Table* is used to choose only *descriptions* for a *language*.

Table 31: Summary of Word Search Tables

Table	Description
<i>Excluded Words Table</i>	Each row in this table is a word excluded from the list of possible <i>keywords</i> and <i>dualkeys</i> . Words are excluded if they are frequently used and are so limited in semantic specificity that they impair rather than enhance searches.
<i>DescWordKey Table</i>	Each row in this table is a word followed by a reference to a <i>Description</i> in which this word appears.
<i>ConcWordKey Table</i>	Each row in this table is a word followed by a reference to a <i>Concept</i> . A <i>Concept</i> is referenced if the word appears anywhere in the combination of the <i>Fully Specified Name</i> with the <i>current valid Preferred Term</i> and <i>Synonyms</i> .

Table	Description
DescDualKey Table	Each row in this table is a six-character <i>string</i> representing the first three letters of a pair of words followed by a reference to a <i>Description</i> in which these two words appear.
ConcDualKey Table	Each row in this table is a six-character <i>string</i> representing the first three letters of a pair of words followed by a reference to a <i>Concept</i> . A <i>Concept</i> is referenced if both words appear anywhere in the combination of the <i>Fully Specified Name</i> with the <i>current valid Preferred Term</i> and <i>Synonyms</i> .

All *keywords* are regarded as case independent and are presented in the word search tables in upper case. Case dependent searching can be applied by appropriately filtering the candidate matches.

4.3.7.4.4 Word Equivalents



The *Word Equivalent* Table is included in the Developer Toolkit of *SNOMED CT*. It supports enhanced searches that take into account semantically similar words such as KIDNEY and RENAL. It also provides commonly used abbreviations. This table can be used by implementers to offer additional search capability in applications without greatly increasing the volume of *synonyms*. It is not intended as a comprehensive dictionary of words. Many searches can be completed without using this table; like the other word search tables, it is completely optional and can be used as an example of a capability that may be customized and extended by *SNOMED CT* implementers.

4.3.7.4.4.1 Word Equivalents Tables - Summary



Table 32: Word Equivalents Table

Key Fields	
WordBlockNumber	A 32-bit <i>integer</i> shared by a set of equivalent words or phrases. The <i>WordBlockNumber</i> links together several rows that have an identical or similar meaning.
WordText	A word, phrase, acronym or abbreviation that is equivalent to the <i>WordText</i> of other rows that share the same <i>WordBlockId</i> .
Data Fields	
WordType	An <i>integer</i> indicating the type of <i>equivalence</i>
WordRole	An <i>integer</i> indicating the usual role of this word. This should be considered if attempting to find a <i>postcoordinated</i> combination of <i>Concepts</i> that matches a phrase.

Chapter

5

5 Release File Specifications



This part of the guide specifies the formats in which *SNOMED CT* is provided to licensees (*IHTSDO affiliates*).

 **Note:** For *SNOMED CT* licensing information and details about the availability of *release files* contact either the *IHTSDO* or the *IHTSDO Member* in your country. Contact details are available on the *IHTSDO* web site: www.ihtsdo.org.

5.1 Release File Formats



Currently, during a transitional period, there are two distinct *Release Formats* for *SNOMED CT*:

- *Release Format 2 (RF2)*: The new standard distribution format for *SNOMED CT*. This was developed in response to extensive feedback on its predecessor and will replace *RF1* during 2012.
- *Release Format 1 (RF1)*: The specification in which *SNOMED CT* has been provided since its first release in 2002. This format will be phased out, but support of applications that require *RF1* format files will be available using a conversion application developed and supplied by the *IHTSDO*.

The key enhancements in *RF2* are:

- More robust and consistent version representation;
- *Reference sets*, provides a more easily extensible and maintainable replacement for *RF1 subsets* and crossmaps;
- Use of an added *hierarchy* to represent metadata about the structure of *SNOMED CT* itself.

Both *Release Formats* represent:

- The components of *SNOMED CT*:
 - *Concepts*
 - *Descriptions*
 - *Relationships*
- Additional *derivatives* that provide standard representations of :
 - Value-sets consisting of a specified set of *concepts* or *relationships*
 - *Cross-Mapping* tables to other codes and classifications.

Both *Release Formats* are provided in:

- Tab-delimited text files;
- Represent character content in accordance with the *Unicode UTF-8* specification;
- Use *SNOMED CT Identifiers* as the permanent *Identifier* of released core components;
- Support *extensions* to the *International Release* using *namespaces* allocated to licensees to denote the provenance of added components and to ensure *Identifier* uniqueness.

5.2 SNOMED CT Editions, Extensions, Releases and Modules



SNOMED CT is delivered as sets of files containing terminology components and derivatives. The format, content and names of the files delivered conform to SNOMED CT specification and guidelines published by the IHTSDO.

- Components represent the content on the terminology.
- The standard SNOMED CT representation for content is as three interrelated files. The Concept file contains unique identifiers for clinical ideas, the Description file links human readable terms with identified concepts and the Relationship file represents associations between identified concepts.
- Derivatives facilitate the effective use of the terminology.
- The standard SNOMED CT representation for derivatives is a consistent but flexible file format, known as the reference set format. Reference sets can be used for a wide range of purposes including subsets, language preferences, ordered lists, hierarchies, annotations and mapping to or from other terminologies, classifications and code systems.

IHTSDO maintains and delivers shared content and derivatives that provide the foundation of the SNOMED CT. This is known as the SNOMED CT International Edition.

IHTSDO also authorizes Members and Affiliates to maintain and deliver additional components and derivatives known as SNOMED CT Extensions.

- Any organization maintaining an Extension needs to have a namespace identifier allocated to them by the IHTSDO. Every component created must be allocated a new permanent SNOMED CT identifier which is in the originating organization's namespace allocated. The namespace identifier of the originating organization forms part of every component identifier allocated.

IHTSDO Members may maintain and deliver additional terminology components and derivatives that adapt the terminology to meet specific national requirements (National Extension).

Examples:

A National Extension may include:

1. translation into the national language or adaption to a national dialect;
2. additional content to support national policy objectives, a national drug dictionary or other specific requirements;
3. derivatives that configure use of SNOMED CT content by specifying subsets of content to be used for particular purposes;
4. derivatives that map other code systems used in that country to or from SNOMED CT.

IHTSDO Affiliates may also maintain and deliver additional terminology components and derivatives that adapt the terminology to meet the needs of a particular organization, customer or software solution (Affiliate Extension).

Examples: An Affiliate Extension may include:

1. additional content enable a health provider organization or clinical specialty group to address its priority use cases;
2. derivatives that configure use of SNOMED CT in ways that reflect the needs of a health provider organization or specialty;
3. derivatives that configure the way SNOMED CT is used or presented to different customers using a particular software applications;
4. derivatives that map local or proprietary code systems to or from SNOMED CT.

The SNOMED CT International Edition can be used without any Extensions. However, a SNOMED CT Extension cannot be used on its own because all Extensions are dependent on the International Edition and some Extensions are also dependent on other Extensions. Therefore, for each Extension there is a corresponding Edition that includes the Extension, the International Edition and any other Extensions on which it depends.

Examples:

1. Healthitia, an IHTSDO Member, produces the Healthitia National Extension. Affiliates in Healthitia use the Healthitia National Edition (the International Edition and the Healthitia National Extension).
2. Anyville Hospital in Healthitia is an IHTSDO Affiliate. It produces its a local Anyville Extension for use in the hospital and this Extension also depends upon the Healthitia National Extension. Therefore, the Anyville Edition consists of the International Edition, the Healthitia National Extension and the Anyville Extension;
3. AcmeGest an international provider of antenatal care systems is an IHTSDO Affiliate and uses the AcmeGest Extension to support some of its specialist functionality. This Extension only depends on the International Edition, therefore the AcmeGest Edition consist of the International Edition and the AcmeGest Extension.
4. The obstetric department of Anyville Hospital in Healthitia uses the AcmeGest system., so it requires the AcmeGest Extension as well as the Anyville Edition. The AcmeGest Extension has a dependency on the International Edition but this should only be resolved once.

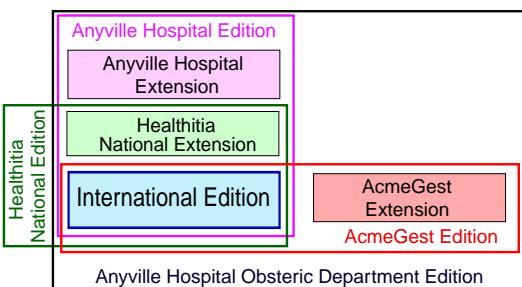


Figure 36: Illustration of the relationships between Extensions and Editions

For use in particular countries or institutions there may be advantages in distribution a complete Edition. However, as the last of example above illustrates, different Editions may share a common dependency on the International Edition. All organizations that maintain Extensions should make their Extension available as a separate set of files, even if they also provide a pre-merged Edition. This allow validation of the constituent parts of the Edition and also supports merges to produce bespoke Editions that combine several Extensions.

The IHTSDO provides regular updates of the SNOMED CT International Edition. Similarly, Members and Affiliates that maintain Extensions will from time to time release updated versions of their Extensions. Therefore,

The SNOMED CT release file specifications define three different release file types:

- Full Release: containing the complete history of every component
- Snapshot Release: containing the current state of every component
- Delta Release (containing only the additions and changes since the previous release)

The International Edition is provided in all three release types. However, as the Snapshot and the Delta can be generated from the Full, the specification only requires that organizations that maintain Extensions provide a Full Release of their Extension.

5.3 Release Format 2 - Introduction



This section describes the revised *release file* structure of SNOMED CT®. This file structure is referred to as *Release Format 2 (RF2)* to distinguish it from the original *Release Format (RF1)* in which SNOMED CT has been distributed since its first release in 2002.

The *Release Format 2* specification is divided into two parts. The *RF2 Core Component Guide* is concerned with the representation of the *Concepts*, *Descriptions* and *Relationships* that contain the primary content of SNOMED CT. The *RF2 - Reference Sets Guide* specifies the common extensible pattern that is used to add additional information related to the core components. It also describes the ways in which this pattern is used to represent essential functionality (such as language specificity, historical status changes and associations) and optional additional functionality (including subsets, *cross mapping* and alternative *navigation hierarchies*).

Files that conformed to the *RF2* specification were available as technology preview during 2010 and early 2011 to enable practical testing of this format. The results of feedback from that testing process have been incorporated into the specification. *RF2* distribution files are available for the current *International Releases* of SNOMED CT and these files also contain the full history of all previous releases. During 2012 *RF2* will become the primary *Release Format* for SNOMED CT. Implementers requiring data in the *RF1* format during a transitional period will be supported by use of conversion application developed by the IHTSDO.

5.4 SNOMED CT - File Naming Conventions



The file naming convention specified in this section applies to all *IHTSDO release files* starting with the January 2010 *International release*.

The specification provides the following benefits:

- A consistent naming convention across the *International edition* and each *National edition*.
- Predictable file naming, providing a stable structure for naming over time between releases.
- A standard way to identify the source country and *namespace* by which a *release file* is owned.
- A consistent versioning mechanism.
- An easy human readable way to identify the content of a file, at a summary level.
- A mechanism for identifying the type of information stored in a *release file* (e.g. documentation, tooling, etc.).
- Guidance on file naming for *release files* in non-English *extensions*.
- Assurance that names will be unique across the *International release* and releases from individual *National release centers* and across separate releases from each center over time.
- An upgrade path, to enable use of the same naming convention with the new *release format (RF2)*, while enabling easy identification of whether a file is in *RF1* or *RF2* format, and avoiding naming clashes.

Quality Assurance checks, to ensure that this naming convention is enforced, will be performed as part of the *International release* process. It is expected that equivalent checks will be performed as part of each *National Release Center's* release process.

 **Note:** Prior to January 2010 other naming strategies were used. Implementers who need to review earlier releases should consult the documentation that accompanied the release that they need to review.

5.4.1 File Naming Convention - Overview



5.4.1.1 General File Naming Pattern



The basic pattern for *SNOMED CT release file* names consists of five elements, each separated by an underscore (" _ ") and followed by a full stop (" .") and a file extension:

Each element in the above structure is described in more detail in subsequent sections.

5.4.1.2 General Naming Rules



The following rules apply generally to all elements of the file name:

- All elements are mandatory and may not have a null value;
- Elements of the file name may only contain alphanumeric characters, with the exception of hyphens (" -") used in connection with *language* codes (see detail for the ContentSubType element below);
- All text should be in US English, except as explicitly allowed below;
- Abbreviations should not be used, except for specified codes or tags;
- The maximum length of a file name (including separators and extension) is 128 characters.

5.4.1.3 Rules for "Readme" Files



Readme files distributed as part of a *SNOMED CT release* have their own specific naming convention, as shown below:

Language is the *language* code for the *language* of the Readme file, as specified below for the ContentSubType element, and the VersionDate corresponds to the version date of the release.

5.4.2 FileType element



5.4.2.1 Description



The FileType element of the filename designates the type and intended use of the *release file*. It consists of a 3-5 letter code and must be lowercase.

5.4.2.2 Rules



Allowable FileType codes are shown in the table below:

Table 33: Allowable File Type Codes

Code	File Type Description
"sct" + <format tag>	Terminology Data File
"der" + <format tag>	Derivative Work Data File
"res" (+ <format tag>)	Implementation Resource Data File
"trs" (+<format tag>)	Implementation Resource Tool
"doc" (+<format tag>)	Documentation
"z" + Code	Archival/Unsupported File (e.g. zsct)

Code	File Type <i>Description</i>
"x" + Code	Test/Beta <i>Release File</i> (e.g. xder)

The allowable file types are described in more detail below:

- **Terminology Data File(" sct")** - the set of data files that make up the *SNOMED CT* terminology. These are:
 - *Concepts table*
 - *Relationships table*
 - *Descriptions table*
 - *Component History table (RF1 only)*;
 - *References table (RF1 only)*;
 - *Identifier table (RF2 only)*.
- **Derivative Work Data File(" der")** - data files that make up a *SNOMED CT* "derivative work" (a product for use in conjunction with *SNOMED CT* that cannot be effectively used without the terminology - such as *subsets* or *Cross-Maps*). Examples of the files within this group include:
 - *Subsets table (RF1 only)*;
 - *SubsetMembers table (RF1 only)*;
 - *CrossMapSets table (RF1 only)*;
 - *CrossMaps table (RF1 only)*;
 - *CrossMapTargets table (RF1 only)*;
 - *Refset table (RF2 only)*.
- **Implementation Resource Data File(" res")** - data files intended to support developers and with the implementation of *SNOMED CT*, but that are not necessarily useful to end-users. Examples from the *current International Release* include:
 - *Canonical table*
 - *Stated Relationships table*
- **Implementation Resource Tool(" tls")** - software tools or other files that do not contain original *SNOMED CT* content (i.e. that is not also held elsewhere in the release), but can be of use to implementers. If such files cannot comply with this naming convention (for example, if some other standard applies), then those files should be distributed as part of a ZIP file archive that does conform to this file naming convention. Examples from the *current International Release* include:
 - Index Generator.
- **Documentation(" doc")** - documents defining *SNOMED CT* standards, policies and guidelines, as well as documentation for files or products included in a *SNOMED CT release*. Most, but not all, files in this group are released in a PDF format.
- **Archival/Unsupported File("zsct", "zder", "zres", "ztls")** - files that are not currently supported or updated, but may be of some use to implementers. These files should only be used with caution and after appropriate review and validation. The letter "z" is inserted in front of the usual FileType code for these files (i.e. "z" + "sct", "der", "res" or "tls"). Examples from the *current International Release* include:
 - *SNOMED 3.5 to SNOMED RT bridge file*;
 - *SNOMED 2 to SNOMED RT bridge file*.
- **Test/Beta Release File("xsct", "xder", "xres", "xtls")** - files distributed as part of a test/beta release, or as a "technology preview". These files should only be used for review and evaluation purposes. The letter "x" is inserted in front of the usual FileType code for these files (i.e. "x" + "sct", "der", "res" or "tls").

5.4.2.3 Format Tags



A *Release Format* tag must be appended at the end of the three-letter FileType code if the file named is dependent on a particular *Release Format* specification. The allowable *Release Format* tags are:

- For files that are part of the *current Release Format*, or applicable only to the *current Release Format*, the number "1" should be appended to the FileType code (e.g. "sct1", "der1", "tls1").
- For files that are part of the *RF2 Release Format*, or applicable only to the *RF2 Release Format*, the number "2" should be appended to the FileType code (e.g. "sct2", "der2", "res2").
- If the file is not specific to either *Release Format*, the three-letter FileType code should be used without a *Release Format* tag (e.g. "res", "tls" or "doc").
- The FileType code for all terminology and *Derivative Work* data files ("sct" or "der") must include a *Release Format* tag ("1" or "2"). For other file types, the *Release Format* tag is optional.

5.4.3 ContentType element



5.4.3.1 Description



The ContentType element of the filename describes the content and purpose of the file. It consists of 2-48 alphanumeric characters in camel case.

5.4.3.2 Rules



The content of this element depends on the first element (FileType) of the filename, as described below:

- **For Data Files("sct", "der" or "res")** - the name of the table contained in the file should be used as the value for the ContentType element. Possible values for the *RF1 Release Format* are:
 - Concepts
 - Descriptions
 - Relationships
 - ComponentHistory;
 - References;
 - Subsets
 - SubsetMembers;
 - CrossMapSets;
 - CrossMaps;
 - CrossMapTargets;
 - TextDefinitions;
 - Canonical;
 - DualKeyIndex;
 - WordKeyIndex;
 - StatedRelationships.

Possible values for the *RF2 Release Format* are:

- Concept
- Description
- Relationship
- Identifier,
- Refset

For Data files where the *ContentType* element is "Refset", the *ContentType* element will also describe the format and content of the *reference set* member file. Each file of *ContentType* "Refset" will hold zero or more additional columns, each having one of the following types:

- Component
- String
- Integer

Lower case "c", "s" and "i" will be used as abbreviations (for *component*, *String* and *Integer* respectively) to describe the format of the additional columns that will be appended to the end of each row in the *Refset* file. These abbreviations will prefix the *ContentType* element, as shown in the examples below:

- **cRefset** - a *Refset* file with one additional column, holding *component* values;
- **ssRefset** - a *Refset* file with two additional columns, both holding *String* values;
- **ciRefset** - a *Refset* file also with two additional columns, the first holding component values and the second holding *integer* values;
- **For ImplementationResource Tools("tls")** - the value of the *ContentType* element may be determined on a case-by-case basis but, in conjunction with the *ContentSubType* element, should be adequate to identify the content and purpose of the file;
- **For Documentation("doc")** - the title of the document, which may be abridged but should not be abbreviated, should be used as the value for the *ContentType* element;
- **For Archival & Test/Beta Files("z"+ code or "x"+ code)** - the value of the *ContentType* element should be determined according to the rules for a normal file of the same type ("sct", "der", "res" or "tls").

5.4.4 ContentSubType element



5.4.4.1 Description



The *ContentSubType* element of the filename provides additional information to describe the content and purpose of the file, including the *language / dialect*, where appropriate. Its format is 2-48 alphanumeric characters in camel case (except for the capitalization rules specified below for *language* code). Hyphen ("") is a permitted character in conjunction with a *language* code, as described below.

5.4.4.2 Rules



The content of this element depends on the first element type (FileType), and these rules are described in more detail below:

- **Data Files("sct", "der" or "res")** - as a result of *RF2*'s state-valid history tracking capability, it is possible to perform a number of different releases of *SNOMED CT* content in the *RF2* format:
 - A "**Full**" release of each file containing every version of every component ever released.
 - A "**Snapshot**" release, containing only the most recent version of every component ever released (both *active* and *inactive components*).
 - A "**Delta**" release, containing only component versions created since the last release. Each component version represents a new component or a change in an existing component.

Each *RF2* *ContentSubType* element must be postfix with a *Release Type* flag with a value of: "**Full**", "**Snapshot**" or "**Delta**". This flag should always appear at the end of the *ContentSubType* element, unless *ContentSubType* includes a *language* code (see below). If a *language* code is present in the *ContentSubType* element, the *Release Type* flag will appear immediately before the *language* code.

In certain cases, an *RF1* data file will be adequately identified by the information appearing in the *ContentType*, so that there is no information that is required to appear in the *ContentSubType* element. In such cases, a placeholder should be used to avoid giving this element a NULL value. The allowable placeholders are:

- "**Core**" - for files that are part of the *SNOMED CT International Release*;
- "**National**" - for files that are part of the *National Release* of an *IHTSDO Member* country; and;

- "**Local**" - for files that are released by an *Affiliate Licensee* or other authorized 3rd party.

These placeholders should not be used in file names for *RF2*-format files.

- **Implementation Resource Tool ("tls")** - the value of this element may be determined on a case-by-case basis but, in conjunction with the *ContentType* element, should be adequate to identify the content and purpose of the file. If appropriate, the element may contain a *status* tag with one of the values described below under Documentation.
- **Documentation ("doc")** - the element should contain at least two components: a *status* tag and a *language* code (see above). Additional components may be added to this element if necessary to fully identify the document. Possible values for the document *status* tag are:
 - "**Current**" - indicates that the document is up-to-date and complete for the *current* release of *SNOMED CT*, as indicated by the *VersionDate* element;
 - "**Draft**" - indicates that the document is a draft version; it may be incomplete and has not been approved in a final version;
 - "**Review**" - indicates that the document has been released for review and comments from *SNOMED CT* users and other stakeholders.
- **Archival & Test/Beta File ("z"+ code or "x"+ code)** - the value of the element should be determined according to the rules for a normal file of the same type ("**sct**", "**der**", "**res**" or "**tls**").

5.4.4.3 Language Usage



For files released as part of a National or local release, and which do not appear in the *SNOMED CT International Release*, the value of the *ContentSubType* element may be given in a *language* other than English, with the following limitation:

- Any of the four sets of defined values for the *ContentSubType* element that are present in the file name may not be translated, but must appear as specified herein. These are: *language* code, *Release Type* flags ("**Full**", "**Snapshot**", "**Delta**"), placeholders ("**Core**", "**National**", "**Local**"), and *status* tags ("**Current**", "**Draft**", "**Review**").

5.4.4.4 Language Codes



Where it is necessary to specify the *language* of a file, a *language* code must be included in the *ContentSubType* element. A *language* code is a *string* identifying the *language* and, if appropriate, the *dialect* of a file, and consists of a code and optionally a sub-code. If a sub-code is present it is separated from the code by a hyphen (" - ").

The code is the two-character *ISO 639-1 language* code. *ISO 639* is the International Standard for "Codes for the representation of names of *languages*". The sub code is a *string* of upper-case letters that represent the *dialect*. This deliberately mirrors the W3C approach and will either be:

- If the *dialect* is general to an entire country, the two-letter *ISO 3166* country code is used. *ISO 3166* is the International Standard for "Codes for the representation of names of countries".
- If *dialects* are used that are less common or not country or *language* linked, the IANA approach is used; this code consists of a *string* of more than two letters. IANA is the Internet Assigned Numbers Authority.

This structure follows Internet conventions. Examples: "**en**" for English, "**es**" for Spanish, "**en-US**" for United States English, "**en-GB**" for British English.

If the *ContentSubType* includes more than one component (e.g. document *status* and a *language* code), the *language* code must be the last component in the *ContentSubType* element and should be preceded by a hyphen (" - ") placed before the *language* code.

5.4.5 Country|Namespace element



5.4.5.1 Description

The Country | Namespace element of the filename helps to identify the organization responsible for developing and maintaining the file. Its format is 2-10 alphanumeric characters consisting of 0, 2 or 3 upper-case letters followed by 0 or 7 digits.

5.4.5.2 Rules



The following rules apply to the content of this element:

- Letters, if present, are either the ISO -3166 2-character country code for an IHTSDO Member country or "INT" for files that are part of the IHTSDO's International Release of SNOMED CT;
- Digits, if present, are a SNOMED CT Namespace Identifier.

Valid combinations are:

- 2 characters only - the file is part of a Member National Release, but not part of a specific Namespace - this combination is not valid for Data Files ("sct", "der" or "res");
- 3 characters only ("INT") - the file is part of the IHTSDO International Release and belongs to the International Namespace;
- 2 characters and 7 digits - the file is part of a Member National Release and belongs to the specified Namespace;
- 3 characters and 7 digits - the file is an optional part of the IHTSDO International Release and belongs to the specified Namespace; or;
- 7 digits only - the file has been developed and released by a 3rd party, identified by the specified Namespace.

5.4.6 VersionDate element



5.4.6.1 Description



The VersionDate element of the filename identifies the SNOMED CT version with which the file is intended to be used. Its format is an 8-digit number in the pattern "YYYYMMDD", in compliance with the ISO 8601 standard.

5.4.6.2 Rules



The following rules apply to the content of this element:

- For Data Files("sct", "der" or "res"), and for Documentation ("doc") with a status tag value of "Current", the value of this element should always be the same as the SNOMED CT version date with which the file is associated.
- For other file types, the VersionDate element will identify the (past) date of the SNOMED CT release for which the file was intended. A file distributed with a past version date has not been updated to reflect changes to SNOMED CT since that date, nor has it been validated as correct or appropriate for current use.

VersionDate refers to the official, published date of a SNOMED CT International Release, or of the National Release of an IHTSDO Member country, and may not always correspond to the actual date of distribution of any particular release.

5.4.7 Extension element



The extension element of the filename identifies the file format (encoding convention) of the file, such as "txt", "pdf" or "zip". It has a format of 1-4 alphanumeric characters.

5.5 Release Format 2 - Core Component Guide



This guide describes *SNOMED CT Release Format 2 (RF2)*, to be used for official production releases of *SNOMED CT*. This format is not mandated for internal terminology development usage or as an interchange mechanism between terminology development systems.

The purpose of *RF2* is to provide a format that is flexible, unambiguous and useful. Its primary aim is to strengthen *SNOMED CT* by providing a format that is simple and stable, while enabling innovation through adaptations to cater for changing requirements.

This specification was developed by harmonizing proposals reviewed by the *IHTSDO Enhanced Release Format Project Group*, including:

- The “Enhanced Release Format Specification” (International Health Terminology Standards Development Organisation. *SNOMED Clinical Terms ® Enhanced Release Format Proposed Specification* , 21 June 2007).
- The “Reference Set Specification” (International Health Terminology Standards Development Organisation. *SNOMED Clinical Terms ® Reference Sets - Proposed Specification* , 31 July 2007).
- The “Alternate Release Format” proposed by NEHTA in coordination with their Australian Affiliates.

5.5.1 General



5.5.1.1 File Naming and Layout



In *RF2*, *release files* will be named predictably and in such a way as to avoid naming clashed between files in the *International release* and *National releases*. The basic pattern for *SNOMED CT release file* names consists of five elements, separated by an underscore ("_"), followed by a full stop (".") and a file extension:

Full details of the file naming convention can be found in the " *SNOMED CT File Naming Convention*" document(see associated documentation).All *release files*:

- are *UTF-8* encoded, tab delimited text files.
- contain a column header row, providing field names for each column within the file. Lower camel case is used for the field names (e.g. *moduleId*, *effectiveTime*).
- use DOS style line termination. Each line is terminated with a carriage return character followed by a line feed character.
- Should have a last line that ends with a line terminator (CR/LF) before the end of file.

5.5.1.2 Field Data Types



The following data types are used in the *release files*:

Table 34: Data Types Used in Release Files

Data Type	Description
<i>SCTID</i>	A <i>SNOMED CT identifier</i> , between 6 and 18 digits long, as described in <i>SCTID Representation</i> .
<i>UUID</i>	Universally Unique <i>Identifier</i> , 128-bit unsigned <i>integer</i>
<i>Integer</i>	32-bit signed <i>integer</i> .

Data Type	Description
String	UTF-8 text of a specified length.
Boolean	Boolean value, represented as one of two possible integer values ('1' for true, '0' for false).
Time	For <i>release files</i> , a time format down to day of the year is used, having an ISO 8601 basic representation of YYYYMMDD. For development interchange formats, an ASCII text field in the ISO 8601 basic format YYYYMMDDThhmmss Z will be used. The time zone will always be UTC, as indicated by the trailing "Z". (e.g. 20080602T223000Z represents 10:30pm June 2 2008 UTC.)

5.5.1.3 Metadata and Enumerated Values



Concept enumerations are used across all *release files*. A concept enumeration simply uses concepts in a metadata hierarchy to represent an enumerated value set rather than using arbitrary integer values directly. A concept enumeration will therefore use an *SCTID* data type.

Non-clinical metadata is separated from the *SNOMED CT* clinical content by holding the two types of data in two separate hierarchies. The concept named | SNOMED CT Model Component |, which is a child of the root concept | SNOMED CT Concept |, contains the metadata model that supports each release.

Underneath the | SNOMED CT Model Component | hierarchy, the | core metadata concept | sub-hierarchy contains concepts that are referenced from fields within other International Release files (the *Concept*, *Description*, *Relationship*, *Identifier* files).

The | foundation metadata concept | sub-hierarchy also sits below the | SNOMED CT Model Component | hierarchy. This sub-hierarchy contains the metadata that supports the extensibility mechanism, and is discussed in more detail in the [Reference Sets Guide](#).

The third and forth sub-hierarchy under | SNOMED CT Model Component | are the | linkage concept | sub-hierarchy, which holds details of relationship types, and the | namespace concept | sub-hierarchy, which holds details of Namespaces.

For more information, see [Concept Enumerations](#).

5.5.1.4 Identification of Source Module



A *moduleId* field, assigned to each component, helps identify the origin of content and dependencies in a release. This enables release centers to compose a unified release from a number of different modules, yet still identify the origin of content within the release. For example, module ids may be used to differentiate *SNOMED CT* International content, Australian Medicines terminology and Pathology content within the Australian National release.

Each component within a *SNOMED CT* release references a *moduleId*. This is the module in which the component is currently maintained. A module is simply a collection of *SNOMED CT components* that are maintained as a unit by a single organization. It is the organization's responsibility to organize the components in each extension that it is responsible for into one or more modules, in a way that best fits its business needs.

5.5.1.5 Meaning of the active field



Each component in RF2 has an associated *active* field, which can take values of true ('1') or false ('0'). The meaning of this flag is described by component type in the following table:

Table 35: Behavior of Active and Inactive Components

Component Type	active value	Description of behavior when most recent row representing a component has the specified active value
<i>Concept</i>	True	<ul style="list-style-type: none"> The <i>Concept</i> is intended for <i>active</i> use. All <i>active Descriptions</i> for which the <i>conceptId</i> refers to this <i>Concept</i> are valid. Visibility of these <i>active Descriptions</i> depends on information contained in applicable RefsetMembers (for example, whether the <i>Description</i> is in a <i>language dialect reference set</i> that is currently enabled in the vendor's system). All <i>active Relationships</i> of which it is the <i>sourceld</i> or <i>destinationId</i> are applicable.
<i>Concept</i>	False	<ul style="list-style-type: none"> The <i>Concept</i> is not intended for <i>active</i> use. However, it remains a valid <i>concept</i> for historical purposes as part of the <i>SNOMED CT</i> commitment to the principle of '<i>concept permanence</i>'. Valid <i>Descriptions</i> of the <i>Concept</i> remain <i>active</i> allowing it to be appropriately viewed in human-readable form. An <i>inactive Concept</i> cannot be the <i>sourceld</i>, <i>destinationId</i> or <i>typeid</i> of an <i>active Relationship</i>.
<i>Description</i>	True	<ul style="list-style-type: none"> The <i>Description</i> contains a <i>Term</i> that is a valid <i>description</i> of the <i>Concept</i> referred to by the <i>conceptId</i>. An <i>active Description</i> may refer to an <i>inactive Concept</i>, in which case the <i>Term</i> provides a valid <i>description</i> of that <i>inactive Concept</i>. Text based searches should (by default) include only <i>active Descriptions</i> that refer to <i>active Concepts</i>.
<i>Description</i>	False	<ul style="list-style-type: none"> The <i>Description</i> is not a valid and the associated <i>Term</i> should no longer be regarded as being associated with the <i>Concept</i> referred to by <i>conceptId</i>.
<i>Relationship</i>	True	<ul style="list-style-type: none"> The <i>Relationship</i> represents a valid association of the type specified by the <i>typeid</i>, between two <i>Concepts</i> referred to by the <i>sourceld</i> and <i>destinationId</i>. An <i>inactive Concept</i> cannot be the <i>sourceld</i>, <i>destinationId</i> or <i>typeid</i> of an <i>active Relationship</i>.
<i>Relationship</i>	False	<ul style="list-style-type: none"> The <i>Relationship</i> is not valid. An <i>inactive Relationship</i> should be ignored as it does not apply. This does not necessarily mean that the association indicated by the <i>Relationship</i> does not apply. The <i>Relationship</i> may be <i>inactive</i> because it is redundant and inferable based on other <i>active Relationships</i>. An <i>inactive Relationship</i> may refer to either <i>active</i> or <i>inactive components</i>.
<i>RefsetMember</i>	True	<ul style="list-style-type: none"> The <i>RefsetMember</i> contains valid information applicable to the <i>component</i> referred to by the <i>referencedComponentId</i>. The <i>component</i> referred to by the <i>referencedComponentId</i> may be <i>active</i> or <i>inactive</i>. An <i>active RefsetMember</i> cannot make an <i>inactive component</i> <i>active</i> but may provide related information that continues to be relevant (e.g. the <i>reason</i> for inactivation).

Component Type	active value	Description of behavior when most recent row representing a component has the specified active value
RefsetMember	False	<ul style="list-style-type: none"> The RefsetMember is not valid. An <i>inactive</i> RefsetMember should be ignored. The information it contains is not applicable to the <i>component</i> referred to by <i>referencedComponentId</i>.

5.5.1.6 History Mechanism



The *effectiveTime* and *active* fields in the *release file* enable the use of a "log style" append-only data model to track all changes to each *component*, providing full traceability. Once released, a row in any of these files will always remain unchanged. Historic data is supplied in the *RF2 release files*, dating back to the first release in *RF1* format in 2002.

In order to change the properties of a *current component* (and, therefore, to create a new version of it), a new row is added to the applicable file, containing the updated fields, with the *active* field set to true and the timestamp in the *effectiveTime* field indicating the nominal date on which the new version was released.

To deactivate a *component*, a new row is added, containing the same data as the final valid version of the *component*, but with the *active* field set to false and the timestamp in the *effectiveTime* field indicating the nominal date of the release in which the final version ceased being valid.

Where editorial policy does not allow a particular property of a *component* to be changed whilst keeping the same *Identifier*, the *component* as a whole is deactivated (as described above), and a new row added with a new id, the effectiveTimeset to the nominal date of the release in which this version of the *component* became valid, and the *active* field set to true.

It is thus possible to see both the *current* values and any historical values of a *component* at any point in time.

Content will not be future dated with respect to the release that it appears in, although a release itself may be released a few days before its nominal release date. Where there is a business requirement for specifying a future activation date for some *components*, this may be modeled using *reference sets*.

The following example demonstrates how the *history mechanism* works on the *Concept file*, but the same rules apply equally well to the *Description*, *Relationship* and *Reference set member* files. In this example, the *descriptions* associated with the *ModuleId* and *DefinitionStatusId* have been shown in place of their *SCTID* values.

A new *concept* (101291009) is added on the 1st July 2007:

Table 36: History Example - Concept Added

Id	effectiveTime	active	moduleId	definitionStatusId
101291009	20070701	1	Module 1	Primitive

In the next release (on 1 st January 2008), the *concept* is moved from |Module 1| to |Module 2|. Because the moduleIdfield is not immutable, the *concept* may be updated simply by adding a new record with the same Id.

Table 37: History Example - Module Change

Id	effectiveTime	active	moduleId	definitionStatusId
101291009	20070701	1	Module 1	Primitive
101291009	20080101	1	Module 2	Primitive

In the next release (on 1st July 2008), the *concept* is changed from being | *Primitive* | to being | *Fully defined* |.

Table 38: History Example - Definition Status Changed

Id	effectiveTime	active	moduleId	definitionStatusId
101291009	20070701	1	Module 1	Primitive
101291009	20080101Z	1	Module 2	Primitive
101291009	20080701	1	Module 2	Fully defined

In the next release (on 1 st January 2009), the *concept* is deactivated:

Table 39: History Example - Concept Made Inactive

Id	effectiveTime	active	moduleId	definitionStatusId
101291009	20070701	1	Module 1	Primitive
101291009	20080101	1	Module 2	Primitive
101291009	20080701	1	Module 2	Fully defined
101291009	20090101	0	Module 2	Fully defined

Note that at no stage in this process are previously written records ever amended. Once a record has been released in a *release file*, it will continue to be released in exactly the same form in future *release files*.

Also, changes are only recorded at the point of release in the *RF2 release files*. If a *component* record is changed a number of times between releases (during an edit and review process), only the most recently amended record will be appended to the *release file*, not individual records showing each separate edit to the released *component*.

5.5.1.7 Release Types



Given the *RF2*'s history tracking capability, it is possible to perform a number of different releases of content:

Table 40: SNOMED CT Release Types

Release Type	Description
Full	The files representing each type of component contain every version of every component ever released.
Snapshot	The files representing each type of component contain one version of every component released up to the time of the snapshot. The version of each component contained in a snapshot is the most recent version of that component at the time of the snapshot.
Delta	The files representing each type of component contain only component versions created since the previous release. Each component version in a <i>delta release</i> represents either a new component or a change to an existing component.

There are valid use cases for each type of *Release Type*. Each *International release* will incorporate all three of these *Release Types*, allowing users to choose the most appropriate format for their needs.

A *full release* will always be available from release centers . Optionally, other *Release Formats* may also be made available. Where out of cycles releases are made, these will follow the same format as standard cycle releases.

5.5.2 Relationships between files



The *relationships* between the records in the *core files* in the *RF2 Release Format* are depicted in the following diagram.

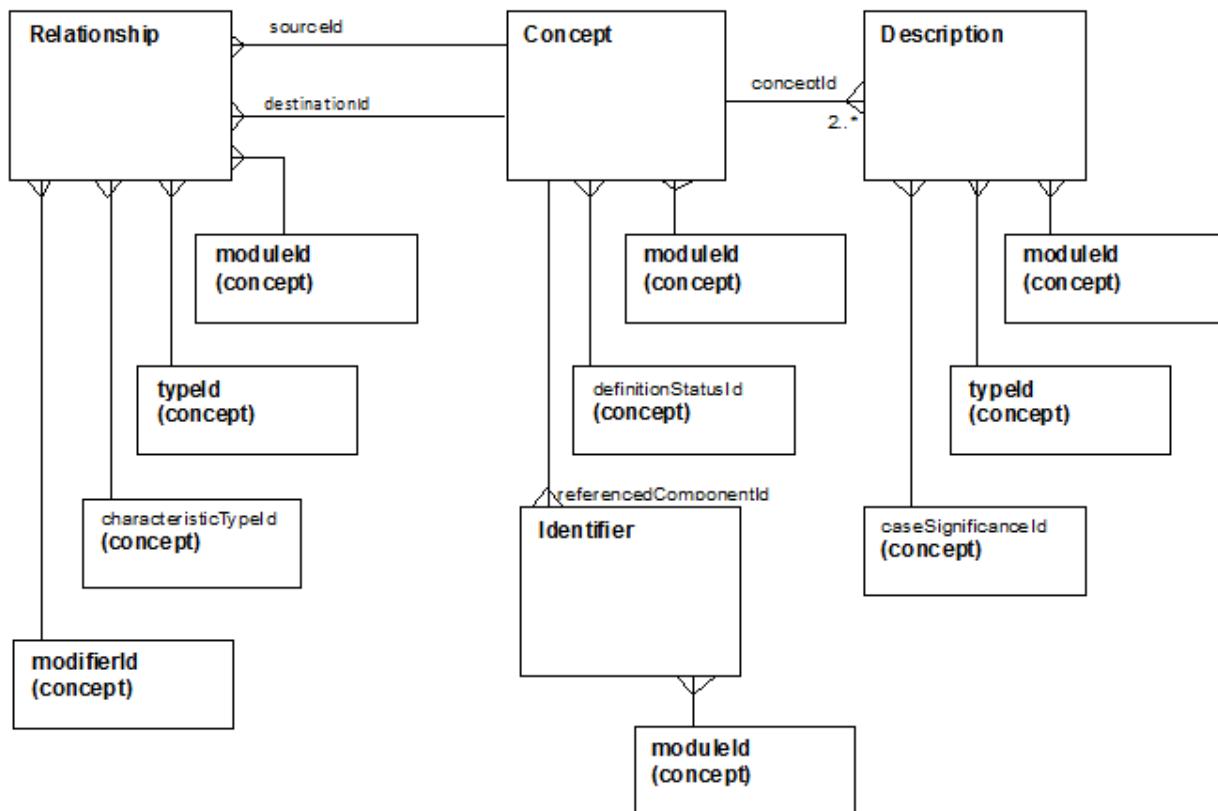


Figure 37: Relationships between files

Each *SNOMED CT concept* is held as a single row in the *Concept file*. Each row represents a clinical *concept*.

Each *concept* has two or more *descriptions* associated with it (at least one *synonym* and at least one *Fully Specified Name*). Each *description* is held as a single row in the *Description file*, and may only refer to a single *concept*.

Each *relationship*, from a source *concept* to a destination *concept*, is held as a single row in the *Relationship file*. The type of each *relationship* is defined by reference to a linkage *concept*, also held within the *Concept file*.

The most basic form of *relationship* is the subsumption *relationship*, identifying that one *concept* is a kind of another *concept*. For example, an *|Outpatient procedure|* *|Is a|* *|Procedure|*. All the *concepts* in *SNOMED CT* form an *|Is a| hierarchy*, with a parent *concept* connected to each child *concept* by an *|Is a| relationship*. In this *hierarchy*, a child *concept* may have more than one parent *concept*. The root of the *hierarchy* is the *|SNOMED CT Concept|*, which has 19 top level *children*, each forming its own *sub-hierarchy*. There are no *|Is a| relationships* that cross from one of these sub-hierarchies to another (e.g. from a *concept* in the *Procedures sub-hierarchy* to a *concept* in the *Substances hierarchy*).

In addition to the *| Is a | relationships*, other *relationship types* are also held within the *Relationship file*, such as *|Finding site|* or *|Laterality|*. *Relationships* types are specified under the *|Linkage| sub-hierarchy* in the *|SNOMED CT Model component| hierarchy*.

5.5.3 File formats



The following sections provide details of the format of the *release files*. An SQL schema, which represents the content of each of these files as a relational table, is provided as part of the *Terminology Service Guide*.

5.5.3.1 Concept file



The *Concept file* holds the clinical *concepts* that make up *SNOMED CT*. A *concept* is given meaning by its *Fully Specified Name*, which is held in the *Description file*. A *concept* may be distinguished from or refined by association with other *concepts* using *relationships*, which are held in the *Relationship file*.

Table 41: Concept file - Detailed Specification

Field	Data type	Immutable	Purpose
id	SCTID	Y	Uniquely identifies the <i>concept</i> .
effectiveTime	Time	N	Specifies the inclusive date at which the component version's state became the then <i>current</i> valid state of the component
active	Boolean	N	Specifies whether the <i>concept</i> 's state was <i>active</i> or <i>inactive</i> from the nominal release date specified by the <i>effectiveTime</i>
moduleId	SCTID	N	Identifies the <i>concept</i> version's module. Set to a <i>descendant</i> of Module within the metadata <i>hierarchy</i> .
definitionStatusId	SCTID	N	Specifies if the <i>concept</i> version is <i>primitive</i> or <i>fully defined</i> . Set to a <i>child</i> of Definition status in the metadata <i>hierarchy</i> .

Only one *concept* record with the same id field is *current* at any point in time. The *current* record will be the one with the most recent *effectiveTime* before or equal to the date under consideration. If the *active* field of this record is false ('0'), then the *concept* is *inactive* at that point in time.

When a *concept* is made *inactive*, the following operations take place:

- A new row is added to the *Concepts* file for the *concept*, with the *active* flag set to *inactive* (as described in the section on the *History Mechanism*);
- All *relationships* that have as source the *concept* to be *retired* will themselves be inactivated by adding a new row to the *Relationship file* for each *relationship*, with the *active* flag set to *inactive*;
- All *active descriptions* associated with the *concept* will remain unchanged unless incorrect for the *concept*;
- Rows will be added as needed to the *Historical Association Reference Set*, to model associations from the *retired concept* to other *concepts*;
- *Active descriptions* that are still associated with the *inactive concept* will be added to the | *Description* inactivation indicator reference set |, with an associated value of | *Concept non-current* |.

5.5.3.2 Description file



The *Description file* holds *descriptions* that describe *SNOMED CT concepts*. A *description* is used to give meaning to a *concept* and provide well-understood and standard ways of referring to a *concept*.

Table 42: Description file - Detailed Specification

Field	Data type	Immutable	Purpose
<i>id</i>	<i>SCTID</i>	Y	Uniquely identifies the <i>description</i> .
<i>effectiveTime</i>	<i>Time</i>	N	Specifies the inclusive date at which the component version's state became the then <i>current</i> valid state of the component
<i>active</i>	<i>Boolean</i>	N	Specifies whether the <i>description</i> 's state was <i>active</i> or <i>inactive</i> from the nominal release date specified by the <i>effectiveTime</i> .
<i>moduleId</i>	<i>SCTID</i>	N	Identifies the <i>description</i> version's module. Set to a <i>child</i> of <i> Module </i> within the metadata <i>hierarchy</i> .
<i>conceptId</i>	<i>SCTID</i>	Y	Identifies the <i>concept</i> to which this <i>description</i> belongs. Set to an <i>Identifier</i> of a <i>concept</i> in the <i> SNOMED CT Concept hierarchy</i> within the <i>Concept file</i> . Note that versions of <i>descriptions</i> and <i>concepts</i> don't belong to each other. Which version of any given <i>description</i> is combined with which version of its owning <i>concept</i> depends on the point in time at which they are accessed.
<i>languageCode</i>	<i>String</i>	Y	Specifies the <i>language</i> of the <i>description</i> text using the two character ISO-639-1 code. Note that this specifies a <i>language</i> level only, not a <i>dialect</i> or country code.
<i>typeId</i>	<i>SCTID</i>	Y	Identifies whether the <i>description</i> is an FSN, <i>Synonym</i> or other <i>description</i> type. This field is set to a <i>child</i> of <i> Descriptiontype </i> in the Metadata <i>hierarchy</i> .
<i>term</i>	<i>String</i>	N	The <i>description</i> version's text value, represented in <i>UTF-8</i> encoding.
<i>caseSignificanceId</i>	<i>SCTID</i>	N	Identifies the <i>concept</i> enumeration value that represents the case significance of this <i>description</i> version. For example, the <i>term</i> may be completely case sensitive, case insensitive, initial letter case sensitive. This field will be set to a <i>child</i> of <i> Case significance </i> within the metadata <i>hierarchy</i> .

Only one *description* record with the same id field will be *current* at any point in time. The *current* record will be the one with the most recent effectiveTime before or equal to the point in time under consideration.

If the *active* field of this record is false ('0'), then the *description* is *inactive* at that point in time. If the *active* field is true ('1'), then the *description* is associated with the *concept* identified by the conceptId field.

The conceptId field, the languageCode field and the typeId field will not change between two rows with the same id, in other words they are immutable. Where a change is required to one of these fields, then the *current* row will be de-activated (by appending a row with the same id and the *active* field set to false) and a new row with a new id will be appended. Only limited changes may be made to the 'term' field, as defined by editorial rules.

Each *concept* will have at least one *active description* with a *typeId* of | *Synonym* | for a given *languageCode* (like "en"). Where a *concept* only has one *active description* with a *typeId* of | *Fully Specified Name* | for a given *language code*, then that *Description* can be taken as the *Fully Specified Name* for that *language* and each of its *dialects*, and need not therefore be explicitly included in *language reference sets* for that *language*. Where a *concept* only has one *active description* with a *typeId* of | *Fully Specified Name* | across all *language codes* within a release, then that *Description* can be taken as the *Fully Specified Name* for all *languages* and *dialects*, and need not therefore be explicitly included in any *language reference sets* in that release.

The *Term* field will be restricted as follows:

- to an overall maximum length of 32Kb;
- to a maximum length, configurable for each *description type* (as defined by the *Description Type reference set* member associated with that *description type* - see the "SNOMED CT Release Format 2 - Reference Set Specifications" document for more details);
- The format of the *term* field (plain text, limited HTML, XHTML, DITA) will also be configurable for each *description type*, using the same mechanism as above;
- Control characters (including TABs, CRs and LFs) will not appear in |Plain text| and |Limited HTML| format types.

5.5.3.3 Relationship file



The *Relationship file* holds one *relationship* per row. Each *relationship* is of a particular type, and has a source *concept* and a destination *concept*. An example of a *relationship* is given below:

| Outpatient procedure | | Is a | | Procedure | where:

- | Outpatient procedure | is the source *concept*;
- | Is a | is the *relationship type concept*; and;
- | Procedure | is the destination *concept*.

Table 43: Relationship file - Detailed Specification

Field	Data type	Immutable	Purpose
id	SCTID	Y	Uniquely identifies the <i>relationship</i> .
effectiveTime	Time	N	Specifies the inclusive date at which the component version's state became the then <i>current</i> valid state of the component
active	Boolean	N	Specifies whether the <i>relationship</i> 's state was <i>active</i> or <i>inactive</i> from the nominal release date specified by the <i>effectiveTime</i> field.

Field	Data type	Immutable	Purpose
<i>moduleId</i>	SCTID	N	Identifies the <i>relationship</i> version's module. Set to a <i>child</i> of <i>module</i> within the metadata <i>hierarchy</i> .
<i>sourceId</i>	SCTID	Y	Identifies the source <i>concept</i> of the <i>relationship</i> version, i.e., the <i>concept</i> the <i>relationship</i> version emanates from. Set to an <i>Identifier</i> of a <i>concept</i> in the SNOMED CT Concept <i>hierarchy</i> within the "Concept" file.
<i>destinationId</i>	SCTID	Y	Identifies the <i>concept</i> that is the destination of the <i>relationship</i> version. Set to an <i>Identifier</i> of a <i>concept</i> in the SNOMED CT Concept <i>hierarchy</i> within the "Concept" file.
<i>relationshipGroup</i>	Integer	Y	Groups together <i>relationship</i> versions that are part of a logically associated <i>relationship group</i> . All <i>active Relationship</i> records with the same <i>relationshipGroup</i> number and <i>sourceId</i> are grouped in this way.
<i>typeId</i>	SCTID	Y	A <i>concept</i> enumeration value from the metadata <i>hierarchy</i> that identifies the semantic type of the <i>relationship</i> version. For example Is a , or associated morphology .
<i>characteristicTypeId</i>	SCTID	Y	A <i>concept</i> enumeration value that identifies the characteristic type of the <i>relationship</i> version (i.e. whether the <i>relationship</i> version is defining, qualifying, etc.) This field is set to a <i>descendant</i> of <i>characteristic type</i> within the metadata <i>hierarchy</i> .
<i>modifierId</i>	SCTID	Y	A <i>concept</i> enumeration value that identifies the type of <i>Description Logic</i> (DL) restriction (some, all, etc.). Set to a <i>child</i> of <i>modifier</i> within the metadata <i>hierarchy</i> .

Only one *relationship* record with the same id field will be *current* at any point in time. The *current* record will be the one with the most recent effectiveTimebefore or equal to the point in time under consideration.

If the *active* field of this record is false ('0'), then the *relationship* is *inactive* at that point in time. If the *active* field is true ('1'), then there is a *relationship* between the SNOMED CT *concepts* identified by *sourceId* and *destinationId*.

The *sourceId*, *destinationId*, *relationshipGroup*, *typeId*, *characteristicTypeId* and *modifierId* will not change between two rows with the same id, in other words they are immutable. Where a change is required to one of these fields, then the *current* row will be de-activated (by appending a row with the same id and the *active* field set to false) and a new row with a new id will be appended.

The *relationshipGroup* field is used to group *relationships* with the same *sourceId* field into one or more logical sets. A *relationship* with a *relationshipGroup* field value of '0' is considered not to be grouped. All *relationships* with the same *sourceId* and non-zero *relationshipGroup* are considered to be logically grouped.

The *relationshipGroup* field will be an unsigned *integer*, and will not be limited to a single digit value. There is no guarantee that they will be assigned sequentially, and the values will not be unique across *concepts*.

The *modifierId* field will initially be set to | Some | to keep compatibility with the *RF1* release. Widening the range of this field to include other values (such as |All|) will in future increase the expressive power of *SNOMED CT*. However, this is likely to come at the cost of an increase in reasoning complexity, leading to potential issues for classification tooling.

Notes:

1. The *modifierId* field has been included at this stage as the *RF2* format is likely to be stable for at least a five year period, without addition or deletion of fields. Within that period it is anticipated that other *modifierId* values will be added. Therefore, although not fully implemented at this stage, this field has been included in the initial *RF2* specification as it represents an integral part of the *Description Logic* used by *SNOMED CT*.
2. Any expansion of *SNOMED CT* to include *relationships* with a *modifierId* set to a value other than | Some | will be discussed with *Members* first and approved by the Technical Committee.

5.5.3.4 Identifier file



This file provides a standardized way of associating alternate *Identifiers* from various schemes with *SNOMED CT components*.

At any point in time, an alternate *Identifier* within a particular scheme will be associated with one and only one *SNOMED CT component*. A *SNOMED CT component* may be associated with zero or more alternate *Identifiers* within a single scheme.

It is important to note that the *SNOMED CT component* and its alternate *Identifiers* all identify precisely the same real-world object.

Table 44: Identifier file - Detailed Specification

Field	Data type	Immutable	Purpose
<i>identifierSchemeId</i>	<i>SCTID</i>	Y	<i>Identifier</i> of the <i>concept</i> enumeration value from the <i>Metadata hierarchy</i> that represents the scheme to which the <i>Identifier</i> value belongs. Set to a descendant of <i>Identifier scheme</i> within the <i>metadata hierarchy</i> .
<i>alternateIdentifier</i>	<i>String</i>	Y	<i>String</i> representation of the alternate <i>Identifier</i> in its native scheme.
<i>effectiveTime</i>	<i>Time</i>	N	Specifies the inclusive date at which the alternate <i>Identifier</i> was associated with the <i>SNOMED CT component</i> .

Field	Data type	Immutable	Purpose
<i>active</i>	<i>Boolean</i>	N	Specifies whether the association was <i>active</i> or <i>inactive</i> from the point in time specified by the <i>effectiveTime</i> .
<i>moduleId</i>	<i>SCTID</i>	N	Identifies the source module that this association was created in. Set to a <i>child</i> of <i> Module </i> within the metadata <i>hierarchy</i> .
<i>referencedComponentId</i>	<i>SCTID</i>	Y	Uniquely identifies the <i>SNOMED CT component</i> with which the alternate <i>Identifier</i> is associated.

Only one record with the same *identifierSchemeId* and *alternateIdentifier* fields will be *current* at any point in time. The *current* record will be the one with the most recent *effectiveTime* before or equal to the point in time under consideration.

If the *active* field of this record is false ('0'), then the association is *inactive* at that point in time. If the *active* field is true ('1'), then there is an identity at that point in time between the *referencedComponentId* (a *SNOMED CT component*) and the *alternateIdentifier* in the scheme identified by *identifierSchemeId*.

5.5.3.5 Transitive Closure History File



The *Transitive Closure* is the complete set of *relationships* between every *concept* and each of its super-type *concepts*, in other words both its parents and *ancestors*. A *Transitive Closure* History file can be generated from the *SNOMED CT* content using scripts provided with each release. The generated file will be of the following format and contain the valid states of the *transitive closure* of each *concept* across all previous releases:

Table 45: Transitive Closure History File - Detailed Specification

Field	Data type	Purpose
<i>subtypeld</i>	<i>SCTID</i>	Id of the <i>concept</i> playing the <i>subtype</i> role. Set to an <i>Identifier</i> of a <i>concept</i> in the <i> SNOMED CT Concept hierarchy</i> within the "Concept" file.
<i>supertypeld</i>	<i>SCTID</i>	Id of the <i>concept</i> playing the super-type role. Set to an <i>Identifier</i> of a <i>concept</i> in the <i> SNOMED CT Concept hierarchy</i> within the "Concept" file.
<i>effectiveTime</i>	<i>Time</i>	Specifies the inclusive date at which the <i>transitive closure</i> record became valid.
<i>active</i>	<i>Boolean</i>	Specifies whether the <i>Identifier</i> version's state was <i>active</i> or <i>inactive</i> from the point in time specified by the <i>effectiveTime</i> .

5.5.4 Extensibility Mechanism



Reference set data structures provide the foundation pieces for *RF2*'s generic extensibility mechanism. These building blocks provide a common foundation for extension owners to build on *SNOMED CT*. They also enable the *Release Format* to support changing requirements.

Conventions applied to the *RF2* files such as field names, field *order* and history tracking have also been applied to the *reference set* specification. This has been done to provide consistency across all components in the *Release Format*.

5.5.4.1 The basic reference set member file format



The basic *reference set* data structure consists of the following fields:

Table 46: Basic Reference Set Data Structure

Field	Data type	Immutable	Purpose
<i>id</i>	<i>UUID</i>	Y	A 128 bit unsigned <i>integer</i> , uniquely identifying the <i>reference set</i> member.
<i>effectiveTime</i>	<i>Time</i>	N	Specifies the inclusive date at which this change becomes effective.
<i>active</i>	<i>Boolean</i>	N	Specifies whether the member's state was <i>active</i> or <i>inactive</i> from the nominal release date specified by the <i>effectiveTime</i> field.
<i>moduleId</i>	<i>SCTID</i>	N	Identifies the member version's module. Set to a <i>child</i> of Module within the metadata <i>hierarchy</i> .
<i>refsetId</i>	<i>SCTID</i>	Y	Uniquely identifies the <i>reference set</i> that this <i>extension</i> row is part of. Set to a <i>descendant</i> of Reference set within the metadata <i>hierarchy</i> .
<i>referencedComponentId</i>	<i>SCTID</i> or <i>UUID</i>	Y	Uniquely identifies the component that this row relates to, thus defining membership of this component in the <i>Reference Set</i> . This field can be set to the <i>Identifier</i> of a record within the <i>Concept</i> , <i>Description</i> , <i>Relationship</i> or <i>Reference Set</i> member file. However, the content of this field can be further restricted for each <i>reference set</i> by the <i>reference set</i> descriptor (see the "SNOMED CT Release Format 2 - Reference Set Specifications" document for more details).

Field	Data type	Immutable	Purpose
Zero or more other fields	<i>SCTID, String, or Integer</i>	N	Optional field
...	<i>SCTID, String, or Integer</i>	N	Optional field

Each *reference set* will be defined as a *concept* in the metadata *hierarchy*.

There will be one *active* row in the above table for each member of the *reference set*. Individual *reference set* members will be uniquely identified using a *UUID*. Each *Reference Set* member will belong to a single *Reference Set* (referred to by the *refsetId* field) and will also reference the member component that belongs to that *reference set* (using the *referencedComponentId* field). The member component may be a *Concept*, *Description*, *Relationship* or a *RefSet* member itself.

Only one *reference set* member record with the same *id* field will be *current* at any point in time. The *current* record will be the one with the most recent *effectiveTime* before or equal to the point in time under consideration.

If the *active* field of this record is false ('0'), then the *reference set* member is *inactive* at that point in time. If the *active* field is true ('1'), then the component referenced by the *referencedComponentId* field is deemed to be a member of the *reference set* identified by the *refsetId* field.

The *refsetId* and *referencedComponentId* fields will not change between two rows with the same *id*, in other words they are immutable. Where a change is required to one of these fields, then the *current* row will be de-activated (by appending a row with the same *id* and the *active* field set to false) and a new row with a new *id* will be appended.

A component may belong to any number of *reference sets* and to each *reference set* more than once. In the latter case, there will be more than one row with the same *refsetId* and *referencedComponentId*, each having different *id* fields, so co-existing at the same time.

5.5.4.2 Extending the basic reference set member file format



The *reference set* member file structure may be extended by addition of one or more fields.

Each of these fields will hold additional values specific to each member. Data types that are supported in the additional columns are:

- *Integer*
- *String*
- *Component* (a reference to a *SNOMED CT component*).

Finer grained interpretation of the values is based on the | *Reference set descriptor* | *reference set*. Further details can be found in the "SNOMED CT Release Format 2 - Reference Set Specifications" document.

The different *Reference Set* patterns that are supported will depend on a documented set of use cases. The supported patterns will expand over time as further use cases are identified.

5.5.5 Metadata hierarchy



As the *release file* formats contain a number of *concept* enumerations, it is necessary to define sets of *concepts* that represent the allowed values. As well as the enumerated values, other metadata supporting the extensibility mechanism and the *concept model* is required.

The *concept* | *SNOMED CT Model Component (metadata)* | is a *subtype* of the *root concept* (| *SNOMED CT Concept* |), and contains the metadata, supporting the release.

The *subtypes* of | *SNOMED CT Model Component (metadata)* | are described in [Table 47](#) and the top three levels of the hierarchy are shown in [Figure 38](#).

Table 47: SNOMED CT Model Component (metadata) (900000000000441003)

Id	Term	Comment
106237007	Linkage concept	<p><i>Concepts</i> that specify</p> <ul style="list-style-type: none"> • Semantic Relationships between concepts (Attribute); and • Asserted associations between statements in a record (Link assertion)
370136006	Namespace concept	<p><i>Concepts</i> that specify the <i>Extension Namespaces</i> allocated by the <i>IHTSDO</i>.</p>
900000000000442005	Core metadata concept	<p><i>Concepts</i> that are referenced from enumerated fields within the <i>International Release files</i> (the <i>Concept, Description, Relationship, Identifier files</i>).</p>
900000000000454005	Foundation metadata concept	<p>The metadata that supports the extensibility mechanism, and is discussed in more detail in the Reference Sets Guide.</p>

- 138875005 | SNOMED CT Concept |
 - ... (*content hierarchies*) ...
 - 900000000000441003 | SNOMED CT Model Component |
 - 106237007 | linkage concept |
 - 246061005 | attribute | ...
 - 416698001 | link assertion | ...
 - 370136006 | namespace concept | ...
 - 900000000000442005 | core metadata concept |
 - 900000000000443000 | module |
 - 900000000000445007 | IHTSDO maintained module | ...
 - 900000000000444006 | definition status |
 - 90000000000073002 | defined |
 - 90000000000074008 | primitive |
 - 900000000000446008 | description type |
 - 9000000000003001 | fully specified name |
 - 90000000000013009 | synonym |
 - 900000000000550004 | definition |
 - 900000000000447004 | case significance |
 - 90000000000017005 | case sensitive |
 - 90000000000020002 | only initial character case insensitive |
 - 900000000000448009 | case insensitive |

- 900000000000449001 | characteristic type |
 - 9000000000000006009 | defining relationship | ...
 - 900000000000225001 | qualifying relationship |
 - 900000000000227009 | additional relationship |
- 900000000000450001 | modifier |
 - 900000000000451002 | some |
 - 900000000000452009 | all |
- 900000000000453004 | identifier scheme |
 - 9000000000002006 | SNOMED CT UUID |
 - 900000000000294009 | SNOMED CT integer identifier |
- 900000000000454005 | foundation metadata concept |
 - 900000000000455006 | reference set |
 - 900000000000456007 | reference set descriptor |
 - 900000000000480006 | attribute value type | ...
 - 900000000000496009 | simple map | ...
 - 900000000000506000 | language type | ...
 - 900000000000512005 | query specification type | ...
 - 900000000000516008 | annotation type | ...
 - 900000000000521006 | association type | ...
 - 900000000000534007 | module dependency |
 - 900000000000538005 | description format |
 - 900000000000457003 | reference set attribute |
 - 900000000000458008 | attribute description |
 - 900000000000459000 | attribute type | ...
 - 900000000000479008 | attribute order |
 - 900000000000491004 | attribute value | ...
 - 900000000000499002 | scheme value |
 - 900000000000500006 | map source concept |
 - 900000000000501005 | map group |
 - 900000000000502003 | map priority |
 - 900000000000503008 | map rule |
 - 900000000000504002 | map advice |
 - 900000000000505001 | map target |
 - 900000000000510002 | description in dialect |
 - 900000000000511003 | acceptability | ...
 - 900000000000514006 | generated reference set |
 - 900000000000515007 | query |
 - 900000000000518009 | annotated component |
 - 900000000000519001 | annotation | ...
 - 900000000000532006 | association source component |
 - 900000000000533001 | association target component |
 - 900000000000535008 | dependency target |
 - 900000000000536009 | source effective time |
 - 900000000000537000 | target effective time |
 - 900000000000539002 | description format | ...

- 900000000000544009 | description length |

Figure 38: SNOMED CT Metadata Hierarchy (2010-07-31)

5.6 Release Format 2- Reference Sets Guide



5.6.1 Introduction



This guide describes the *reference set* specifications released as part of the *SNOMED CT Release Format 2*. This format is not mandated for internal terminology development usage or as an interchange mechanism between terminology development systems.

The purpose of *RF2* is to provide a format that is flexible, unambiguous and useful. Its primary aim is to strengthen *SNOMED CT* by providing a format that is simple and stable, while enabling innovation through adaptations to cater for changing requirements.

This format specification was developed by harmonizing proposals reviewed by *IHTSDO Enhanced Release Format Project Group*, including:

- Enhanced *Release Format Specification* (International Health Terminology Standards Development Organisation. *SNOMED Clinical Terms ® Enhanced Release Format Proposed Specification*, 21 June 2007).
- *Reference Set Specification* (International Health Terminology Standards Development Organisation. *SNOMED Clinical Terms ® Reference Sets - Proposed Specification*, 31 July 2007).
- Alternate *Release Format* proposed by NEHTA in coordination with their Australian Affiliates.

5.6.1.1 Associated Quality Measures



Although the definition of quality measures to monitor the implementation of this standard do not fall under the scope of this guide, they will be covered by the documentation covering the QA and *Release* process for the *Workbench*.

5.6.1.2 Separation of Reference Sets into Release files



Separation of *reference sets* into files may be done in a number of ways. Each *reference set* will have a particular structure for the optional fields that are appended to each member. For example, a simple *reference set* will have no additional fields; a *CSI reference set* will have three additional fields - the first a *component*, the second a *String*, and the third an *Integer*. There must be at least one *reference set* member file for each different *reference set* structure, as defined above. *Reference sets* may be further split, if required, by the owner of the *reference sets*. The naming conventions for the *reference set* files provide more detail on the naming convention to be used in this case (see the " *SNOMED CT File Naming Convention*" document).

Each *reference set* file will have a header row containing field names for each of the columns. The names of the standard fields will be the field names as detailed in the " *SNOMED CT Release Format 2 - Data Structures Specification*" document.

In *release files* the additional fields will be named in accordance with the relevant row in " *Reference Set Descriptor reference set*".

 **Note:** When imported by an application the *release file* names of additional attributes may be substituted by a more generic name to allow *reference sets* with a similar pattern of additional fields to be represented

in a single table. In this case the "*Reference Set Descriptor reference set*" provides a link to the original name of the field in the distribution file.

5.6.2 Reference Set Specifications



This section first details how *reference sets* themselves are described in a machine readable form, using a set of | *Reference set descriptor*| member records (called a Descriptor, for short). It then describes a number of standard *reference set patterns*. Each of these patterns is also described in a machine readable form using a set of | *Reference set descriptor*| member records (called a Descriptor Template, for short). Each pattern may be used to define a number of *reference sets*. At the end of the section, a number of individual *reference sets* are described that do not conform to a particular pattern.

In each subsection, each *reference set* or *reference set pattern* is described in turn:

- The purpose of each *reference set* is first described;
- The format of the *reference set* member record is detailed in a table;
- The metadata supporting the *reference set* is described;
- The machine readable *reference set descriptor* member records for the *reference set pattern* (the Descriptor Template, for short) are then shown;
- Examples of usage are given, providing example Descriptors, where appropriate.

The first *reference set* to be described is the *reference set descriptor*. Subsequent sections describe a number of *reference set patterns*.

5.6.2.1 Overview

5.6.2.1.1 Descriptors, Descriptor Templates and Patterns



The purpose of the | *Reference set descriptor*| is to describe the format of all other *reference sets* that may be included in a release. A Descriptor held within the | *Reference set Descriptor*| describes the referencedComponentId field and the additional fields for the *reference sets* it describes. Each field is described using a *concept* in the metadata. The type of each field is also described in the same way.

Patterns allow a number of different types of *reference set* to be defined, each of which will conform to the specified pattern, having the same *release file* format. The file format of each *reference set pattern* is described by a Descriptor Template. This Descriptor Template describes the format and number of additional fields held against members of *reference sets* conforming to the pattern, and provides an envelope within which those additional fields may be further refined for each *reference set* conforming to the pattern. The Descriptor Template for each pattern is provided in the section describing that pattern.

Each defined *reference set* that conforms to a pattern will have its own Descriptor, that describes its own specific properties, and although *reference set* field types must still conform to the Descriptor Template for the pattern, each field type may be further constrained using data sub-types specified in the metadata *hierarchy*. This provides some level of *refinement* to the *constraints* that may be applied to a *reference set* conforming to a particular pattern.

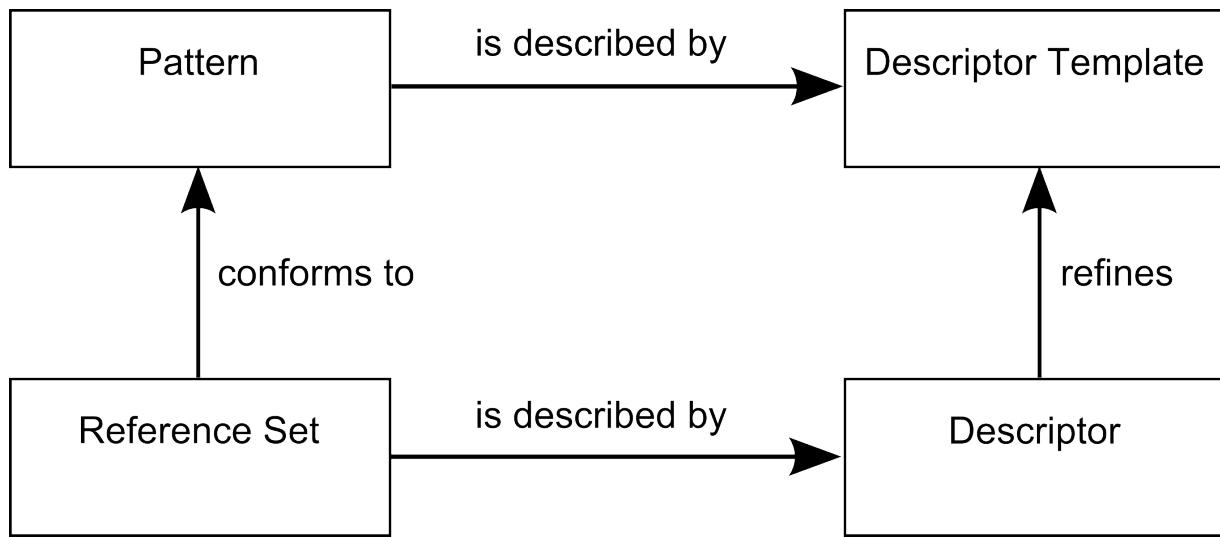


Figure 39: Graphical view of Relationships between patterns, reference sets, Descriptor Templates and Descriptors

5.6.2.1.2 Patterns and Use Cases



The next table summarizes the use cases for *reference sets* (one per row) that are described in the following sections, and shows which *reference set* patterns are used in each case:

Table 48: Reference Set Use Case Summary

		Patterns							
Use cases	Attribute value type (C)	Simple map (S)	Complex map type (IISSSC)	Language type (C)	Query Specification type CCS)	Annotation type (S)	Associated type (C)	Ordered type (IC)	Simple type
Refinability of relationships	*								
ICD-10 mapping	*	*							
Inactivation indicator	*								
CVT3 map		*							
SNOMEDID map			*						
Language dialect				*					
Language dialect with context					*				
Intension reference set specification						*			
Image annotation							*		
Short annotation							*		
Descriptive annotation								*	
Reason for inactivation								*	
RF1 Subset representation									*

 **Note:** The letters shown after each pattern indicate the type and number of additional fields held against each member of a *reference set* conforming to that pattern, where 'C' is short for *component*, 'S' is short for *String* and 'I' is short for *Integer*.

 **Example:** Reference sets conforming to the | *Attribute value type* | (C) pattern will have one additional field held against each member, a component reference; reference sets conforming to the | *Simple type* | pattern will have no additional fields held against each member.

5.6.2.1.3 Metadata Supporting Reference Sets



Reference sets are described by *concepts* under the | *Reference set* | *sub-hierarchy*.

- 900000000000455006 | reference set |
 - 900000000000456007 | reference set descriptor |
 - 900000000000480006 | attribute value type | ...
 - 900000000000496009 | simple map | ...
 - 900000000000506000 | language type | ...
 - 900000000000512005 | query specification type | ...
 - 900000000000516008 | annotation type | ...
 - 900000000000521006 | association type | ...
 - 900000000000534007 | module dependency |
 - 900000000000538005 | description format |

Figure 40: Reference Sets in the Metadata Hierarchy

Values that can be used within *reference set* fields are described in the | *Reference set attribute* | *sub-hierarchy*.

- 900000000000457003 | reference set attribute |
 - 900000000000458008 | attribute description |
 - 900000000000459000 | attribute type | ...
 - 900000000000479008 | attribute order |
 - 900000000000491004 | attribute value | ...
 - 900000000000499002 | scheme value |
 - 900000000000500006 | map source concept |
 - 900000000000501005 | map group |
 - 900000000000502003 | map priority |
 - 900000000000503008 | map rule |
 - 900000000000504002 | map advice |
 - 900000000000505001 | map target |
 - 900000000000510002 | description in dialect |
 - 900000000000511003 | acceptability | ...
 - 900000000000514006 | generated reference set |
 - 900000000000515007 | query |
 - 900000000000518009 | annotated component |
 - 900000000000519001 | annotation | ...
 - 900000000000532006 | association source component |
 - 900000000000533001 | association target component |
 - 900000000000535008 | dependency target |
 - 900000000000536009 | source effective time |
 - 900000000000537000 | target effective time |
 - 900000000000539002 | description format | ...

- 900000000000544009 | description length |

Figure 41: Reference Set Attributes Metadata Hierarchy

The way that each of the *concepts* shown in this metadata *hierarchy* is used is described in each of the following sections.

5.6.2.1.4 Naming Conventions for Reference Sets



National Release Centres and others may create additional *reference sets*. A *namespace* is required to create a new *reference set*, as each *reference set* is defined by a *Concept*. The *Concept's FSN* and a *Synonym* are used to name the *reference set*. Where a new *reference set* is created against an existing pattern, then the following naming convention should be used (where the text "My particular" should be replaced by the name of the *reference set*):

Attribute value type reference set (pattern)

FSN = My particular *attribute value reference set* (*foundation metadata concept*)

PT = My particular *reference set*

Simple Map type reference set (pattern)

FSN = My particular *simple map reference set* (*foundation metadata concept*)

PT = My particular *simple map*

Complex Map type reference set (pattern)

FSN = My particular *complex map reference set* (*foundation metadata concept*)

PT = My particular *complex map*

Language type reference set (pattern) - for a *Language Refset*

FSN = English - ISO 639-1 code 'en' *language reference set* (*foundation metadata concept*)

PT = English

Language type reference set (pattern) - for a *Dialect RefSet*

FSN = GB English *language reference set* (*foundation metadata concept*)

PT = GB English

Query specification type reference set (pattern)

FSN = My particular *query specification reference set* (*foundation metadata concept*)

PT = My particular *query specification reference set*

Annotation type reference set (pattern)

FSN = My particular *annotation reference set* (*foundation metadata concept*)

PT = My particular *annotation reference set*

Association type reference set (pattern)

FSN = My particular *association reference set* (*foundation metadata concept*)

PT = My particular *association reference set*

5.6.2.2 Reference Set Descriptor



5.6.2.2.1 Purpose

The *Reference Set Descriptor* *reference set* is used to describe the format of all other *reference sets* that are included in a release. The data type and meaning of the referenced component and each additional field within each *reference set* is described by this *reference set*.

Reference set descriptor can be used to define

- The order of appearance of additional attributes (other than those mandatory for a *reference set*);
- The name and purpose of the additional attributes;
- The data types for the additional attributes.

This allows for a *reference set* to be validated using the metadata embedded within the *reference set descriptor* in the following ways:

- the data type of its attributes may be validated against the data type declared in the *reference set descriptor*;
- the column order can be checked against the *reference set descriptor*.

5.6.2.2.2 Reference Set Data Structure



The *Reference Set Descriptor* *reference set* is a CCI (*component - component - Integer*) *reference set* and has the following format.

Table 49: Descriptor Reference Set - Data Structure

Field	Data type	Purpose
<i>id</i>	<i>UUID</i>	
<i>effectiveTime</i>	<i>Time</i>	
<i>active</i>	<i>Boolean</i>	
<i>moduleId</i>	<i>SCTID</i>	
<i>refsetId</i>	<i>SCTID</i>	Indicates that this row is part of a "reference set descriptor". Set to the 900000000000456007 Reference set descriptor reference set (foundation metadata concept)
<i>referencedComponentId</i>	<i>SCTID</i>	Identifies the <i>reference set</i> (or type of <i>reference set</i>) that is specified by this descriptor. Set to a descendant of 900000000000455006 reference set (foundation metadata concept) in the metadata hierarchy.
<i>attributeDescription</i>	<i>SCTID</i>	Specifies the name of an attribute that is used in the <i>reference set</i> to which this descriptor applies. Set to a descendant of 900000000000457003 Reference set attribute (foundation metadata concept) in the metadata hierarchy, that describes the additional attribute extending the <i>reference set</i> .

Field	Data type	Purpose
<i>attributeType</i>	SCTID	Specifies the data type of this attribute in the <i>reference set</i> to which this descriptor applies. Set to a <i>descendant</i> of 900000000000459000 attribute type (foundation metadata concept) in the metadata <i>hierarchy</i> , that describes the type of the additional attribute extending the <i>reference set</i> .
<i>attributeOrder</i>	Integer	Specifies the position of this attribute in the <i>reference set</i> to which this descriptor applies. A zero value identifies the <i>referencedComponentId</i> within the <i>reference set</i> . Other values specify an additional attributes by its position relative to the <i>referencedComponentId</i> . An unsigned <i>integer</i> , providing an ordering for the additional attributes extending the <i>reference set</i> .

At least one row must exist for each *reference set* included in a release. This row must have an *attributeOrder* value of '0' and an *attributeType* of 'component type' (or one of its descendants). The *referencedComponentId* then describes the member (or referenced component) of the *reference set*.

One additional row will exist to describe each additional attribute that extends the *reference set* under consideration.

Creation of a *Refset* descriptor is mandatory when authoring a *refset* in the International release or an NRC extension.

Otherwise, creation of a *Refset* descriptor is optional. Where a *refset* descriptor is not defined for a *reference set*, then the closest *ancestor* of the *reference set* that has a *refset* descriptor can be used when validating *reference set* member records.

5.6.2.2.3 Metadata



The following metadata in the |Foundation metadata concept | *hierarchy* supports the *reference set* descriptor *reference set*.

The *Reference Set Descriptor Reference Set* is specified by the 900000000000456007 | Reference set descriptor | *concept* in the metadata hierarchy.

- 900000000000441003 | SNOMED CT Model Component |
 - 900000000000454005 | Foundation metadata concept |
 - 900000000000455006 | Reference set |
 - 900000000000456007 | Reference set descriptor |

Figure 42: Reference Set Descriptor Concept in the Metadata Hierarchy

Values in the *Reference Set* are populated from:

- 900000000000454005 | Foundation metadata concept |
 - 900000000000457003 | Reference set attribute |

- 900000000000458008 | Attribute description |
 - 900000000000459000 | Attribute type |
 - 900000000000460005 | Component type |
 - 900000000000461009 | Concept type component |
 - 900000000000462002 | Description type component |
 - 900000000000463007 | Relationship type component |
 - 900000000000464001 | Reference set member type component |
- 900000000000465000 | String |
 - 900000000000466004 | Text |
 - 900000000000467008 | Single character |
 - 900000000000468003 | Text < 256 bytes |
- 900000000000469006 | URL |
 - 900000000000470007 | HTML reference |
 - 900000000000471006 | Image reference | ...
- 900000000000474003 | UUID |
 - 900000000000475002 | Time |
- 900000000000476001 | Integer |
 - 900000000000477005 | Signed integer |
 - 900000000000478000 | Unsigned integer |
- 900000000000460005 | Component type | ...
 - 900000000000465000 | String | ...
 - 900000000000476001 | Integer | ...
- 900000000000479008 | Attribute order |
 - 900000000000491004 | Attribute value | ...

Figure 43: Reference Set Attribute Metadata Hierarchy**5.6.2.2.4 Descriptor**

The following table shows the | Reference set descriptor| active member entries for the | Reference set descriptor| itself:

Table 50: Descriptor for the Descriptor Template Reference Set

refsetId	referencedComponentId	attributeDescription	attributeType	attributeOrder
Reference set descriptor	Reference set descriptor	Reference set	Concept type component	0
Reference set descriptor	Reference set descriptor	Reference set Attribute	Concept type component	1
Reference set descriptor	Reference set descriptor	Attribute type	Concept type component	2

refsetId	referencedComponentId	attributeDescription	attributeType	attributeOrder
Reference set descriptor	Reference set descriptor	Attribute order	Unsigned integer	3

In the above and following examples, *descriptions* have been used in place of *component Identifiers* for clarity.

Within a particular release, *attributeOrders* will be contiguous for a particular referencedComponentId within a *reference set* descriptor.

5.6.2.3 Simple Reference Set

5.6.2.3.1 Purpose



The Simple Reference Set provides allows a set of *components* to be specified for inclusion or exclusion from a specified purpose. This type of Reference Set can be used to enumerate the members of a simple subset or *value set*.

5.6.2.3.2 Reference Set Data Structure



A Simple *reference set* does not have any addition fields.

Table 51: Simple Reference Set - Data Structure

Field	Data type	Purpose
id	UUID	A 128 bit unsigned <i>integer</i> , uniquely identifying the <i>reference set</i> member.
effectiveTime	Time	Specifies the inclusive date at which this change becomes effective.
active	Boolean	Specifies whether the member's state was <i>active</i> or <i>inactive</i> from the nominal release date specified by the <i>effectiveTime</i> field.
moduleId	SCTID	Identifies the member version's module. Set to a <i>child</i> of Module within the metadata <i>hierarchy</i> .
refsetId	SCTID	Set to a <i>child</i> of Simple type in the metadata <i>hierarchy</i> .
referencedComponentId	SCTID	A reference to the SNOMED CT component to be included in the <i>reference set</i> .

5.6.2.3.3 Metadata



Simple References Sets are *subtypes* of #sctid# | Simple Reference set | in the metadata hierarchy.

- 900000000000441003 | SNOMED CT Model Component |
 - 900000000000454005 | Foundation metadata concept |
 - 900000000000455006 | Reference set |

- #sctid# | Simple Reference set |

Figure 44: Simple Reference Sets in the Metadata Hierarchy**5.6.2.3.4 Descriptor Template**

One *reference set* descriptor member will be required for each instance of the Simple type *Reference Set*.

The table below holds the Descriptor Template for the | Simple type | *reference set* pattern:

Table 52: Descriptor Template for Simple Reference Sets

refsetId	referencedComponentId	attributeDescription	attributeType	attributeOrder
Reference set descriptor	Simple type	Referenced component	component type	0

5.6.2.4 Ordered Reference Set

This *reference set* pattern allows a collection of components to be defined, and optionally given a priority ordering, and split into a number of sub-groups, if necessary.

It can be used to represent *Navigation*, *Duplicate terms*, *Realm concept*, *Realm relationship* and *Context concept reference sets*.

5.6.2.4.2 Reference Set Data Structure

An *Integer component reference set* is used to support the *Ordered type* pattern.

Table 53: Ordered Reference Set - Data Structure

Field	Data type	Purpose
id	UUID	A 128 bit unsigned <i>integer</i> , uniquely identifying the <i>reference set</i> member.
effectiveTime	Time	Specifies the inclusive date at which this change becomes effective.
active	Boolean	Specifies whether the member's state was <i>active</i> or <i>inactive</i> from the nominal release date specified by the <i>effectiveTime</i> field.
moduleId	SCTID	Identifies the member version's module. Set to a <i>child</i> of Module within the metadata <i>hierarchy</i> .
refsetId	SCTID	Set to a <i>child</i> of Ordered type in the metadata <i>hierarchy</i> .
referencedComponentId	SCTID	A reference to the <i>SNOMED CT component</i> being tagged with a value.

Field	Data type	Purpose
<i>order</i>	<i>Integer</i>	The priority <i>order</i> of this item in the set, where a value of '1' is the highest priority, and higher values are of lower priority. A value of '0' is not allowed.
<i>linkedTold</i>	SCT/ <i>ID</i>	<p>This field either enables members of the <i>reference set</i> -to be linked into a number of sub-groups, or enables a number of <i>children concepts</i> to be linked to a single parent concept.</p> <p>To link members into a sub-group, all components in the same sub-group should reference the component in the group that has an <i>order</i> of '1' (i.e. - the highest priority component). Therefore, all components that have the same <i>linkedTold</i> value will be in the same sub-group.</p> <p>To link a number of <i>children concepts</i> to a single parent concept, one member record should exist per <i>child</i>, with the <i>referencedComponentId</i> field referencing the parent and this field referencing the <i>child concept</i>. The <i>order</i> field is then used to order the <i>children concepts</i> under the parent concept.</p> <p>For members that are not linked in either of the above ways, this field should be set to '0'.</p>

5.6.2.4.3 Metadata



The following metadata in the "Foundation metadata concept" hierarchy supports this *reference set*:

- 900000000000454005 | Foundation metadata concept |
 - 900000000000455006 | Reference set |
 - #sctid# | Ordered reference set | ...

Figure 45: Ordered References Sets in the Metadata Hierarchy

5.6.2.4.4 Descriptor Template



One group of *reference set* descriptor members will be required for each instance of the |Ordered type| *Reference Set*.

The table below holds the Descriptor Template for the | Ordered type | *reference set* pattern:

Table 54: Descriptor Template for Ordered Reference Sets

refsetId	referencedComponentId	attributeDescription	attributeType	attributeOrder
Reference set descriptor	Ordered type	Referenced component	component type	0

refsetId	referencedComponentId	attributeDescription	attributeType	attributeOrder
Reference set descriptor	Ordered type	Order	Unsigned integer	1
Reference set descriptor	Ordered type	Linked to	component type	2

5.6.2.5 Attribute Value Reference Set

5.6.2.5.1 Purpose



This *reference set* pattern allows a value from a specified range to be associated with a component.

5.6.2.5.2 Reference Set Data Structure



A *component reference set* will be used to support the *attribute value* pattern.

Table 55: Attribute Value Reference Set - Data Structure

Field	Data type	Purpose
id	UUID	A 128 bit unsigned <i>integer</i> , uniquely identifying the <i>reference set</i> member.
effectiveTime	Time	Specifies the inclusive date at which this change becomes effective.
active	Boolean	Specifies whether the member's state was <i>active</i> or <i>inactive</i> from the nominal release date specified by the <i>effectiveTime</i> field.
moduleId	SCTID	Identifies the member version's module. Set to a <i>child</i> of Module within the metadata <i>hierarchy</i> .
refsetId	SCTID	Set to a <i>child</i> of Attribute value type in the metadata <i>hierarchy</i> .
referencedComponentId	SCTID	A reference to the <i>SNOMED CT component</i> being tagged with a value.
valueld	SCTID	Set to a grandchild of "Attribute value".

5.6.2.5.3 Metadata



The following metadata in the "Foundation metadata concept" hierarchy supports this *reference set*:

- 900000000000454005 | Foundation metadata concept |
 - 900000000000455006 | Reference set |
 - 900000000000480006 | Attribute value type |

- 900000000000488004 | Relationship refinability reference set |
- 900000000000489007 | Concept inactivation indicator reference set |
- 900000000000490003 | Description inactivation indicator reference set |
- 900000000000547002 | Relationship inactivation indicator reference set |

Figure 46: Attribute Value Reference Sets in the Metadata Hierarchy**5.6.2.5.4 Descriptor Template and example Descriptors**

One group of *reference set* descriptor members will be required for each type of *attribute value reference set*.

The table below holds the Descriptor Template for the | Attribute value type | *reference set* pattern:

Table 56: Descriptor Template for Attribute Value Reference Sets

refsetId	referencedComponentId	attributeDescription	attributeType	attributeOrder
Reference set descriptor	Attribute value type	Referenced component	component type	0
Reference set descriptor	Attribute value type	Attribute value	Concept type component	1

Table 57: Descriptor for the ICD-10 map category reference set

refsetId	referencedComponentId	attributeDescription	attributeType	attributeOrder
Reference set descriptor	ICD-10 map category reference set	Referenced component	Reference set member type component	0
Reference set descriptor	ICD-10 map category reference set	ICD-10 map category value	Concept type component	1

Table 58: Descriptor for the Relationship refinability reference set

refsetId	referencedComponentId	attributeDescription	attributeType	attributeOrder
Reference set descriptor	Relationship refinability reference set	Referenced component	Relationship type component	0
Reference set descriptor	Relationship refinability reference set	Refinability value	Concept type component	1

Table 59: Descriptor for the Concept inactivation indicator reference set

refsetId	referencedComponentId	attributeDescription	attributeType	attributeOrder
Reference set descriptor	Concept inactivation indicator reference set	Referenced component	Concept Type component	0
Reference set descriptor	Concept inactivation indicator reference set	Concept inactivation value	Concept Type component	1

Table 60: Descriptor for the Description inactivation indicator reference set

refsetId	referencedComponentId	attributeDescription	attributeType	attributeOrder
Reference set descriptor	Description inactivation indicator reference set	Referenced component	Description type component	0
Reference set descriptor	Description inactivation indicator reference set	Description inactivation value	Concept type component	1

Similar members will also exist for the *relationship* inactivation indicator and the *reference set* inactivation indicator.

5.6.2.5.5 Attribute Value Refset Examples



A

5.6.2.5.5.1 Example: Relationship Refinability

**Table 61: Example Attribute Value Reference Set- Relationship refinability**

refsetId	referencedComponentId	valueId
Relationship refinability reference set	A SNOMED CT Relationship	Not refinable
Relationship refinability reference set	A SNOMED CT Relationship	Optional refinability
Relationship refinability reference set	A SNOMED CT Relationship	Mandatory refinability

A *relationship* (referenced by the *referencedComponentId* field) can be associated with an enumeration concept (a child of the | *Refinability* value| concept in the metadata hierarchy), held in the *valueId* field.

5.6.2.5.5.2 Example: Inactivation indicator

**Table 62: Example Attribute Value Reference Set - Inactivation Indicator**

refsetId	referencedComponentId	valueId
Concept inactivation indicator reference set	Concept 1	Duplicate
Description inactivation reference set	Description 1	Concept non-current

For the purposes of the above example, we will assume that *Description 1* is an *active description* linked to *Concept 1*, which is *inactive* (this is not shown in the table). The *reference set* members indicate that the reason for *Concept 1*'s inactivation was because it was a duplicate *concept* and *Description 1* is a valid *description* on the *inactive concept*.

5.6.2.6 Simple Map Reference Set**5.6.2.6.1 Purpose**

This *reference set* pattern supports simple maps between *SNOMED CT concepts* and values in alternate coding schemes. No constraints are put on the number of coding schemes supported, the number of codes within a particular scheme mapped to by a single *SNOMED CT concept* or the number of *SNOMED CT concepts* mapping to a particular code. However, it is expected that this *reference set* will be primarily used when there is a reasonably close "one-to-one" mapping between *SNOMED CT concepts* and the alternate coding scheme.

5.6.2.6.2 Reference set data structure

A *String reference set* will be used to support simple maps.

Table 63: Simple Map Reference Set - Data Structure

Field	Data type	Purpose
id	UUID	A 128 bit unsigned <i>integer</i> , uniquely identifying the <i>reference set member</i> .
effectiveTime	Time	Specifies the inclusive date at which this change becomes effective.
active	Boolean	Specifies whether the member's state was <i>active</i> or <i>inactive</i> from the nominal release date specified by the <i>effectiveTime</i> field.
moduleId	SCTID	Identifies the member version's module. Set to a <i>child</i> of <i> Module </i> within the metadata <i>hierarchy</i> .
refsetId	SCTID	Set to a <i>child</i> of the <i> Simple map type </i> in the metadata <i>hierarchy</i> . This identifies the alternate scheme that is being mapped to.

Field	Data type	Purpose
<i>referencedComponentId</i>	SCTID	A reference to the SNOMED CT concept being mapped.
<i>mapTarget</i>	String	The value of the code in the alternate scheme being mapped to.

5.6.2.6.3 Metadata



The following metadata *hierarchy* supports this *reference set*:

- 900000000000454005 | Foundation metadata concept |
 - 900000000000455006 | Reference set |
 - 900000000000456007 | Reference set descriptor | ...
 - 900000000000480006 | Attribute value type | ...
 - 900000000000496009 | Simple map |
 - 900000000000497000 | CTV3 simple map |
 - 900000000000498005 | SNOMED RT ID simple map |

Figure 47: Simple Map Reference Sets in the Metadata Hierarchy

5.6.2.6.4 Descriptor Template and example Descriptors



One group of *reference set* descriptor members will be required for each type of simple map *reference set*.

The table below holds the Descriptor Template for the simple map *reference set* pattern:

Table 64: Descriptor Template for Simple Map Reference Sets

refsetId	referencedComponentId	attributeDescription	attributeType	attributeOrder
Reference set descriptor	Simple map	Referenced component	Concept type component	0
Reference set descriptor	Simple map	Scheme Value	String	1

Table 65: Descriptor Template for CTV3 Simple Map Reference Sets

refsetId	referencedComponentId	attributeDescription	attributeType	attributeOrder
Reference set descriptor	CTV3 simple map	Referenced component	Concept type component	0
Reference set descriptor	CTV3 simple map	Scheme value	String	1

5.6.2.6.5 Simple Map Refset Examples



Table 66: Example rows from CTV3 and SNOMED RT - Simple Map Reference Sets

refsetId	referencedComponentId	mapTarget
CTV3 simple map	10006000	72710
CTV3 simple map	100060003	XU014
SNOMED RT ID simple map	10006000	P1-A68A3
SNOMED RT ID simple map	100060003	C-D1777

5.6.2.7 Complex and Extended Map Reference Sets

5.6.2.7.1 Purpose



These *reference set* patterns support maps where each *SNOMED CT concept* may map to one or more codes in a *target scheme*. A Complex Map Reference Set supports the general set of mapping data required to enable a *target code* to be selected at run-time from a number of alternate codes. It allows *target code* selection to be supported by inclusion of sets of machine readable rules and/or by inclusion of human readable advice. The Extended Map Reference Set adds an additional field to allow categorization of maps.

5.6.2.7.2 Reference Set Data Structure



An IISSSC (*Integer - Integer - String - String - component*) reference set will be used to support complex maps:

Table 67: Complex and Extended Map Reference Sets - Data Structures

Field	Data type	Purpose
id	UUID	A 128 bit unsigned <i>integer</i> , uniquely identifying the <i>reference set member</i> .
effectiveTime	Time	Specifies the inclusive date at which this change becomes effective.
active	Boolean	Specifies whether the member's state was <i>active</i> or <i>inactive</i> from the nominal release date specified by the <i>effectiveTime</i> field.
moduleId	SCTID	Identifies the member version's module. Set to a <i>child</i> of Module within the metadata <i>hierarchy</i> .
refsetId	SCTID	Set to one of the <i>children</i> of the Complex map type <i>concept</i> in the metadata <i>hierarchy</i> .
referencedComponentId	SCTID	A reference to the <i>SNOMED CT concept</i> being mapped.

Field	Data type	Purpose
<i>mapGroup</i>	<i>Integer</i>	An <i>integer</i> , grouping a set of complex map records from which one may be selected as a <i>target code</i> . Where a <i>SNOMED CT concept</i> maps onto 'n' <i>target codes</i> , there will be 'n' groups, each containing one or more complex map records.
<i>mapPriority</i>	<i>Integer</i>	Within a group, the <i>mapPriority</i> specifies the <i>order</i> in which complex map records should be checked. Only the first map record meeting the run - time selection criteria will be taken as the <i>target code</i> within the group of alternate codes.
<i>mapRule</i>	<i>String</i>	A machine-readable rule, (evaluating to either 'true' or 'false' at run-time) that indicates whether this map record should be selected within its <i>mapGroup</i>
<i>mapAdvice</i>	<i>String</i>	Human-readable advice, that may be employed by the software vendor to give an end-user advice on selection of the appropriate <i>target code</i> from the alternatives presented to him within the group.
<i>mapTarget</i>	<i>String</i>	The <i>target code</i> in the scheme to be mapped onto.
<i>correlationId</i>	<i>StId</i>	A <i>child</i> of <i> Map correlation value </i> in the metadata <i>hierarchy</i> , identifying the correlation between the <i>SNOMED CT concept</i> and the <i>target code</i> .
<i>The following additional field only applies to Extended Cross Map Reference Sets</i>		
<i>mapCategoryId</i>	<i>StId</i>	Identifies the <i>SNOMED CT concept</i> in the metadata hierarchy which represents the <i>MapCategory</i> for the associated map member. The categories vary for different <i>target code</i> systems, each set of categories is represented by a <i>subtype</i> of <i>609331003 Map category value </i> . For example in the case of <i>ICD-10</i> the individual category values are <i>subtypes</i> of: <i>447634004 ICD-10 Map category value </i> .

Values for the *mapGroup* field will be allocated on a sequential basis (for each *refsetId* and *referencedComponentId* combination) starting from '1', but are not necessarily sequential, as groups may be created and removed during a mapping process that may straddle several releases. For maps where each *SNOMED CT concept* only maps to at most one of a group of alternate *target codes*, the *mapGroup* field should be set to '1'.

Values for the mapPriority field will be allocated on a sequential basis (within each map group) starting from '1'. For maps that do not require run - time alternatives, the mapPriority field should be set to '1'.

The mapRule and mapAdvice fields enable run - time selection (within vendor's software) from a number of alternative map records within a mapGroup. Where alternatives are not required, these fields should be set to null. Where complex maps are required, either, both or neither of these fields may be populated.

Where both fields are populated, and a vendor's system is capable of processing a machine readable rule, this should take priority over the human readable advice. Where neither field is populated, a vendor's system should allow the end-user to select the appropriate *target code* from the alternates.

5.6.2.7.3 Metadata



The following metadata supports this *reference set*:

- 900000000000454005 | Foundation metadata concept |
- 900000000000455006 | Reference set |
- 447250001 | Complex map type reference set | ...
- 609331003 | Extended map type reference set | ...

Figure 48: Complex Map References Sets in the Metadata Hierarchy

5.6.2.7.4 Descriptor Template



The table below holds the Descriptor Template for this *reference set* pattern:

Table 68: Descriptor Template for Complex Map Reference Sets

refsetId	referencedComponentId	attributeDescription	attributeType	attributeOrder
Reference set descriptor	Complex map type reference set	Map source concept	Concept type component	0
Reference set descriptor	Complex map type reference set	Map group	Unsigned integer	1
Reference set descriptor	Complex map type reference set	Map priority	Unsigned integer	2
Reference set descriptor	Complex map type reference set	Map rule	String	3
Reference set descriptor	Complex map type reference set	Map advice	String	4
Reference set descriptor	Complex map type reference set	Map target	String	5
Reference set descriptor	Complex map type reference set	SNOMED CT source code to target map code correlation value	Concept type component	6

Table 69: Descriptor Template for Extended Map Reference Sets

refsetId	referencedComponentId	attributeDescription	attributeType	attributeOrder
Reference set descriptor	Extended map type reference set	Map source concept	Concept type component	0
Reference set descriptor	Extended map type reference set	Map group	Unsigned integer	1
Reference set descriptor	Extended map type reference set	Map priority	Unsigned integer	2
Reference set descriptor	Extended map type reference set	Map rule	String	3
Reference set descriptor	Extended map type reference set	Map advice	String	4
Reference set descriptor	Extended map type reference set	Map target	String	5
Reference set descriptor	Extended map type reference set	SNOMED CT source code to target map code correlation value	Concept type component	6
Reference set descriptor	Extended map type reference set	Map category value	Concept type component	7

5.6.2.7.5 Mapping rule specifications



The specific grammar and content of the rules for resolving complex mapping cases depends on the nature of the *target code* system or classification. In general, each map is accompanied by a rule which is tested against other data and can be evaluated to return one of the following values:

- **True** - in which case the map target applies;
- **False** - in which case the map target does not apply;
- **Indeterminate** - in cases where there is insufficient accessible data to determine whether the map target applies. In this case manual resolution of the map using the map advice provided will be required.

The mapping rules assume access to a number of variables, that can be bound to appropriate attributes in the vendor's system information model. These include the age and gender of the patient and information about coexisting situations (e.g. records of other disorders, procedures or events in the same patient record).

Detailed definitions of the mapping rules used forms part of individual specifications for maps to particular *target code* systems and classifications. This will initially be provided separately and will accompany the release of the relevant mapping files. For example, the set of rules used for mapping to *ICD-10* are currently included in a document released with those maps.

5.6.2.7.6 Extended Map Refset Example



The following example uses simplified tokens to represent identifiers and omits the *effectiveTime*, *active* and *moduleId* columns. It also combines the *mapRule* and *mapAdvice* columns and illustrates the content of these rules with an informal representation of the type of rules and advice that may be used.

Table 70: Example row from an ICD-10 Complex Map Reference Set

id	refsetId	referenced ComponentId	map Group	map Priority	mapRule (and mapAdvice)	mapTarget	correlationId	map Category
uid1	ICD-10 complex map	conceptId-A	5	1	If X true, then	P	Narrow to broad map	Map of source concept is context dependent
uid2	ICD-10 complex map	conceptId-A	5	2	Otherwise	Q	Narrow to broad map	Map of source concept is context dependent
uid3	ICD-10 complex map	conceptId-A	6	1	If Y true, then	R	Narrow to broad map	Map of source concept is context dependent
uid4	ICD-10 complex map	conceptId-A	6	2	Otherwise	S	Narrow to broad map	Map of source concept is context dependent

The table holds four *ICD-10* map records, mapping from *SNOMED CT* concept A to two *ICD-10* target codes (represented by the two groups 5 and 6).

Human readable map advice indicates that the *SNOMED CT* concept A should map to both:

- *ICD-10* code "P", if X is true; but otherwise to *ICD-10* code "Q"; and to;
- *ICD-10* code "R", if Y is true; but otherwise to *ICD-10* code "S".

 **Note:** Use of an Extended Map Reference Set pattern, including the mapCategoryId field, removes the requirement for a an additional Map Category Reference Set which was used in preview releases of the *ICD-10* maps.

5.6.2.8 Language Reference Set

5.6.2.8.1 Purpose



This *reference set* pattern supports the creation of sets of *descriptions* for one or more *dialects* of a *language*, perhaps for use within a particular context.

The structure allows one *dialect reference set* to be based on another. In this case, instead of repeating all members of the parent *reference set*, a *child* need instead only specify additional *descriptions* for membership, and exclude those members of the parent *reference set* that are not acceptable as *descriptions*.

5.6.2.8.2 Reference set data structure



A *component reference set* will be used to support *language reference sets*.

Table 71: Language Reference Set - Data Structure

Field	Data type	Purpose
id	<i>UUID</i>	A 128 bit unsigned <i>integer</i> , uniquely identifying the <i>reference set</i> member.

Field	Data type	Purpose
<i>effectiveTime</i>	<i>Time</i>	Specifies the inclusive date at which this change becomes effective.
<i>active</i>	<i>Boolean</i>	Specifies whether the member's state was <i>active</i> or <i>inactive</i> from the nominal release date specified by the <i>effectiveTime</i> field.
<i>moduleId</i>	<i>SCTID</i>	Identifies the member version's module. Set to a <i>child</i> of <i> Module </i> within the metadata <i>hierarchy</i> .
<i>refsetId</i>	<i>SCTID</i>	A <i>descendant</i> of <i> Language type </i> in the metadata <i>hierarchy</i>
<i>referencedComponentId</i>	<i>SCTID</i>	A reference to the <i>Description</i> included in the <i>language reference set</i> .
<i>acceptabilityId</i>	<i>SCTID</i>	A <i>descendant</i> of "Acceptability" in the metadata <i>hierarchy</i> .

Within each *language reference set*, there must be at most one *Description* (for each included *concept*) with a *typeid* of *| Fully Specified Name |*. Additionally, there must be one and only one *Description* (for each included *concept*) with a *typeid* of *| Synonym |* that has an *acceptabilityId* field (within the *reference set*) of *|Preferred|*.

5.6.2.8.3 Metadata



The following metadata supports this *reference set*:

- 900000000000454005 | Foundation metadata concept |
 - 900000000000455006 | Reference set |
 - 900000000000456007 | Reference set descriptor |
 - 900000000000480006 | Attribute value type | ...
 - 900000000000496009 | Simple map | ...
 - 900000000000506000 | Language type |
 - 900000000000507009 | English |
 - 900000000000508004 | GB English |
 - 900000000000509007 | US English |

Figure 49: Language References Sets in the Metadata Hierarchy

The immediate *children* of *| Language type|* will be *languages*. This level may be used to represent the "correct" *language*, where a *language authority* exists. In most cases, however, this level is likely to be empty.

5.6.2.8.4 Descriptor Template



One group of *reference set descriptor* members will be required for each *language reference set*.

The table below holds the Descriptor Template for the *language reference set* pattern:

Table 72: Descriptor Template for Language Reference Sets

refsetId	referencedComponentId	attributeDescription	attributeType	attributeOrder
Reference set descriptor	Language type	Description in dialect	Description type component	0
Reference set descriptor	Language type	Acceptability	Concept type component	1

The table below holds the Descriptor for the "GB English" reference set.

Table 73: Descriptor for the GB English Language reference set

refsetId	referencedComponentId	attributeDescription	attributeType	attributeOrder
Reference set descriptor	GB English	Description in dialect	Description type component	0
Reference set descriptor	GB English	Acceptability	Concept type component	1

5.6.2.8.5 Example usage

**Table 74: Example rows from a general GB English and a specialist GB Language Reference Sets**

refsetId	referencedComponentId	acceptabilityId	active
GB English	Appendicectomy	Preferred	1
GB English	Excision of appendix	Acceptable	1
US English	Excision of appendix	Acceptable	1
US English	Appendectomy	Preferred	1

In the above example, | Excision of appendix | is acceptable in both US and GB English. However, | Appendectomy | is preferred in US English and | Appendicectomy | is preferred in GB English.

 **Note:** As the example is only a sample of the Language Reference Set it is not possible to determine whether the GB and US *preferred terms* are acceptable as *synonyms* in the other *dialect*. However, any *Description* which is not referenced by an active row in the relevant language *reference set* is regarded as unacceptable (i.e. not a valid *synonym* in the language or *dialect*).

5.6.2.9 Query Specification Reference Set

5.6.2.9.1 Purpose



Query specification *reference sets* allows a serialised query grammar to be associated with a *reference set* to enable the generation of its members. The *query* is run against the full content of *SNOMED CT* to produce another *reference set*.

The resulting *reference set* may be of any one of the general references set types subject to expressivity of the query language. A fully expressive *query language* must be able to associate any number of optional fields with a member of the resulting *reference set* based on rules specified in the query.

5.6.2.9.2 Reference Set Data Structure



A *String* *reference set* may be used to support serialised *query* specifications in the *RF2 Release Format*.

Table 75: Query Specification Reference Set - Data Structure

Field	Data type	Purpose
id	UUID	A 128 bit unsigned <i>integer</i> , uniquely identifying the <i>reference set</i> member.
effectiveTime	Time	Specifies the inclusive date at which this change becomes effective.
active	Boolean	Specifies whether the member's state was <i>active</i> or <i>inactive</i> from the nominal release date specified by the <i>effectiveTime</i> field.
moduleId	SCTID	Identifies the member version's module. Set to a <i>child</i> of Module within the metadata <i>hierarchy</i> .
refsetId	SCTID	A <i>child</i> of Query specification type in the metadata <i>hierarchy</i> .
referencedComponentId	SCTID	The <i>reference set</i> for which members are to be generated.
query	String	The serialised <i>query</i> that can be used to (re-)generate the <i>reference set</i> members.

5.6.2.9.3 Metadata



The following metadata in the "Foundation metadata concept" *hierarchy* supports this *reference set*:

- 900000000000454005 | Foundation metadata concept |
 - 900000000000455006 | Reference set |
 - 900000000000512005 | Query specification type |

- 90000000000513000 | Simple query specification |

Figure 50: Hierarchy of Foundation metadata concept**5.6.2.9.4 Descriptor Template**

One group of *reference set* descriptor members will be required for each type of *query*.

The table below holds the Descriptor Template for the *query specification reference set* pattern:

Table 76: Descriptor Template for Query Specification Reference Sets

refsetId	referencedComponentId	attributeDescription	attributeType	attributeOrder
Reference set descriptor	Query specification type	Generated <i>reference set</i>	Concept type component	0
Reference set descriptor	Query specification type	Query	String	1

Table 77: Descriptor for the "CS query specification" reference set

refsetId	referencedComponentId	attributeDescription	attributeType	attributeOrder
Reference set descriptor	CS query Specification	Generated <i>reference set</i>	Concept type component	0
Reference set descriptor	CS query Specification	Query	String	1

5.6.2.9.5 Example Usage

In the example below, "serialised *query 1*" is a text *string* that can be used to generate members for *Reference set 1*, which is a simple member *reference set* (without any additional fields within its member records).

Table 78: Example rows from Query Specification Reference Set

refsetId	referencedComponentId	query
Simple query specification	Reference set 1	"serialised <i>query 1</i> "
CS query specification	Reference set 2	"serialised <i>query 2</i> "

"Serialised *query 2*", however, must also generate a *component id* and a *String* value for each *reference set* member that it generates for *Reference set 2*. In this case, *Reference set 2* is a CS *reference set*.

5.6.2.9.6 Query language specification

The specification of the *query language* has yet to be defined / selected, but it should be capable of:

- Selecting *concepts* using primary fields, subsumption testing, *relationships*, *relationship groups*, set operators (*union*, *intersection*, *excludes*), and lexical *query*;
- Selecting *descriptions*, *relationships* and *reference sets* using similar mechanisms;
- Calculation of values for the *reference set's* extended fields. Identifying the version of the syntax and any *language* syntax variations.
- Expressing ref set *query* definitions for terminologies other than *SNOMED CT*. The syntax should not assume that the only target is *SNOMED CT*, it should allow at least for ICDx, LOINC, ICPC, and local vocabularies, particularly lab related.

The definition of the *query language* is outside the scope of this specification.

5.6.2.10 Annotation Reference Set

5.6.2.10.1 Purpose



Annotation *reference sets* pattern allow *strings* to be associated with components for any specified purpose.

5.6.2.10.2 Reference Set Data Structure



A *String* *reference set* is used to support *annotations*.

Table 79: Annotation Reference Set - Data Structure

Field	Data type	Purpose
<i>id</i>	<i>UUID</i>	A 128 bit unsigned <i>integer</i> , uniquely identifying the <i>reference set</i> member.
<i>effectiveTime</i>	<i>Time</i>	Specifies the inclusive date at which this change becomes effective.
<i>active</i>	<i>Boolean</i>	Specifies whether the member's state was <i>active</i> or <i>inactive</i> from the nominal release date specified by the <i>effectiveTime</i> field.
<i>moduleId</i>	<i>SCTID</i>	Identifies the member version's module. Set to a <i>child</i> of <i> Module </i> within the metadata <i>hierarchy</i> .
<i>refsetId</i>	<i>SCTID</i>	A <i>child</i> of <i> Annotation type </i> in the metadata <i>hierarchy</i> .
<i>referencedComponentId</i>	<i>SCTID</i>	A reference to the component to be annotated
<i>annotation</i>	<i>String</i>	The <i>annotation</i> to attach to the component.

5.6.2.10.3 Metadata



The following metadata in supports this *reference set*:

- 900000000000454005 | Foundation metadata concept |
 - 900000000000455006 | Reference set |
 - 900000000000516008 | Annotation type |

- 90000000000517004 | Associated image |

Figure 51: Annotation References Sets in the Metadata Hierarchy**5.6.2.10.4 Descriptor Template**

One group of *reference set* descriptor members will be required for each *annotation reference set*.

The table below holds the Descriptor Template for the *annotation reference set* pattern:

Table 80: Descriptor Template for Annotation Reference Sets

refsetId	referencedComponentId	attributeDescription	attributeType	attributeOrder
Reference set descriptor	Annotation type	Annotated component	Concept type component	0
Reference set descriptor	Annotation type	Annotation	String	1

The *attributeType* for the *Annotation* field can be any *descendant* of the "*String*" concept in the metadata *hierarchy*. This *hierarchy* is described in more detail under the "*Reference set descriptor*" section.

The table below holds the Descriptor for the "Associated image" *annotation reference set*, which allows URLs to be associated with *concepts*:

Table 81: Descriptor for "Associated image" Annotation Reference Sets

refsetId	referencedComponentId	attributeDescription	attributeType	attributeOrder
Reference set descriptor	Associated image	Annotated component	Concept type component	0
Reference set descriptor	Associated image	Image	URL	1

Note that in the table above, the | URL | concept is a *descendant* of | String | concept in the metadata.

5.6.2.10.5 Example Usage

This table holds example entries for this *reference set*.

Table 82: Example of "Associated image" Annotation Reference Set

refsetId	referencedComponentId	Annotation
Associated image	Concept 1	"http://example.com/picture.jpeg"
Associated image	Concept 2	"http://example.com/picture.gif"

In the above example, the two URLs have been used to annotate two *SNOMED CT concepts*.

It is not recommended that this mechanism be used to annotate *concepts* with text that may require translation to other *languages*. Instead, such text should be included under an appropriate *description* type within the *Description file*.

5.6.2.11 Association Reference Set

5.6.2.11.1 Purpose



Association *reference sets* represent unordered associations of particular types between components.

5.6.2.11.2 Reference Set Data Structure



A *component reference set* will be used to support associations.

Table 83: Association Reference Set - Data Structure

Field	Data type	Purpose
<i>id</i>	<i>UUID</i>	A 128 bit unsigned <i>integer</i> , uniquely identifying the <i>reference set</i> member.
<i>effectiveTime</i>	<i>Time</i>	Specifies the inclusive date at which this change becomes effective.
<i>active</i>	<i>Boolean</i>	Specifies whether the member's state was <i>active</i> or <i>inactive</i> from the nominal release date specified by the <i>effectiveTime</i> field.
<i>moduleId</i>	<i>SCTID</i>	Identifies the member version's module. Set to a <i>child</i> of <i> Module </i> within the metadata <i>hierarchy</i> .
<i>refsetId</i>	<i>SCTID</i>	A <i>descendant</i> of <i> Association type </i> in the metadata <i>hierarchy</i> .
<i>referencedComponentId</i>	<i>SCTID</i>	A reference to the source component of the association.
<i>targetComponentId</i>	<i>SCTID</i>	A reference to the destination component of the association.

5.6.2.11.3 Metadata



The following metadata in the "Foundation metadata concept" *hierarchy* supports this *reference set*:

- 900000000000455006 | Reference set |
 - 900000000000521006 | Association type |
 - 900000000000522004 | Historical association |
 - 900000000000523009 | POSSIBLY EQUIVALENT TO association reference set |
 - 900000000000524003 | MOVED TO association reference set |
 - 900000000000525002 | MOVED FROM association reference set |
 - 900000000000526001 | REPLACED BY association reference set |
 - 900000000000527005 | SAME AS association reference set |

- 900000000000528000 | WAS A association reference set |
- 900000000000529008 | SIMILAR TO association reference set |
- 900000000000530003 | ALTERNATIVE association reference set |
- 900000000000531004 | REFERS TO concept association reference set |

Figure 52: Association Reference Sets in the Metadata Hierarchy**5.6.2.11.4 Notes on usage**

Each member of a | Historical association | reference set represents a Reference from an *inactive component* to other equivalent or related *components* that were current in the *Release Version* in which that *component* was inactivated.

Each | Historical association | reference set holds *Relationships* of a different nature between the *components*. The |Historical association| reference sets contains associations:

- from each *inactive description* to one or more other *Descriptions* that are current in the *release Version* in which the *description* was inactivated;
- from each inactive *reference set* for which there is a current replacement to the replacement *reference set*;
- from an *inactive description* to a *concept* that is current in the *Release Version* in which the *description* was inactivated, and which is correctly described by the Term of the *inactive description*;
- From each *inactive concept* to one or more *concepts* that replace it.

The *component* identified by the *targetComponentId* must be an instance of the same class of *component* as the *component* identified by the *referencedComponentId* for all |Historical association| reference sets apart from the |REFERS TO concept association reference set|.

Within the |REFERS TO concept association reference set|, the *referenced ComponentId* field must be a *description* and the *targetComponentId* must be a *concept*.

The *targetComponentId* is used differently in the |MOVED TO association reference set|. In this case, the *targetComponentId* does not refer directly to a replacement *component*, but rather to the namespace to which the *component* was moved to. The *targetComponentId* actually refers to the *concept* that represents the namespace. This approach is used since the organization sourcing the *component* may not always be able to determine the precise reference that is applicable in the receiving organization (namespace). Thus the responsibility for these references lies with the new responsible (receiving) organization .

5.6.2.11.5 Descriptor Template and Descriptor examples

One group of *reference set descriptor* members is required used to represent each association type *reference set*.

The table below holds the Descriptor Template for the | Association type | reference set pattern:

Table 84: Descriptor Template for Association Reference Sets

refsetId	referencedComponentId	attributeDescription	attributeType	attributeOrder
Reference set descriptor	Association type	Association source component	component type	0
Reference set descriptor	Association type	Association target component	component type	1

The table below holds the Descriptor for the |SAME AS association reference set|. Members of this *reference set* identify a target *component* that is an identical duplicate of the source *component*.

Table 85: Descriptor for | SAME AS association reference set |

refsetId	referencedComponentId	attributeDescription	attributeType	attributeOrder
Reference set descriptor	SAME AS association reference set	Association source component	component type	0
Reference set descriptor	SAME AS association reference set	Association target component	component type	1

5.6.2.11.6 Example Usage - Replaced by

The following table holds example entries for the | Replaced by | reference set.

Table 86: Example rows from Replaced by Reference Set

refsetId	referencedComponentId	targetComponent
REPLACED BY association reference set	Concept 1	Concept 2
REPLACED BY association reference set	Concept 3	Concept 4

In this example, the associations describe that *Concept 1* has been replaced by *Concept 2* and *Concept 3* has been replaced by *Concept 4*.

5.6.2.11.7 Example Usage - Refers to Concept

The following table holds example entries for the "Refers to *Concept*" reference set.

Table 87: Example rows from Refers to Concept Reference Set

refsetId	referencedComponentId	targetComponent
REFERS TO concept association reference set	Desc1	Concept 3
REFERS TO concept association reference set	Desc2	Concept 4

In this example, the associations identify that *Concept 3* is correctly described by the *Term* of the *inactive Description*, Desc1 and *Concept 4* is correctly described by the *Term* of the *inactive Description*, Desc2.

5.6.2.12 Module Dependency Reference Set**5.6.2.12.1 Purpose**

The Module Dependency reference set represents dependencies between module versions.

5.6.2.12.2 Reference Set Data Structure

A String -String reference set will be used to support module dependencies.

Table 88: Module Dependency Reference Set - Data Structure

Field	Data type	Purpose
id	UUID	A 128 bit unsigned <i>integer</i> , uniquely identifying the <i>reference set member</i> .
effectiveTime	Time	Specifies the inclusive date at which this change becomes effective.
active	Boolean	Specifies whether the member's state was <i>active</i> or <i>inactive</i> from the nominal release date specified by the <i>effectiveTime</i> field.
moduleId	SCTID	Identifies the member version's module. Set to a <i>child</i> of Module within the metadata <i>hierarchy</i> .
refsetId	SCTID	A reference to the Module dependency <i>concept</i> in the metadata <i>hierarchy</i> .
referencedComponentId	SCTID	A reference to the module that this module is dependent on, a <i>descendant</i> of Module in the metadata <i>hierarchy</i> .
sourceEffectiveTime	String	The effective time of the source module. This allows a specific module version to be selected as having a dependency. The <i>effectiveTime</i> must match exactly.
targetEffectiveTime	String	The effective time of the target module. This allows a specific module version to be selected as being the subject of a dependency. The <i>effectiveTime</i> must match exactly.

5.6.2.12.3 Metadata



The following metadata in the "Foundation metadata concept" *hierarchy* supports this *reference set*:

- 900000000000454005 | Foundation metadata concept |
 - 900000000000455006 | Reference set |
 - 900000000000534007 | Module dependency |

Figure 53: Module Dependency Reference Set in the Metadata Hierarchy

Each component within a *SNOMED CT release* references a *moduleId*. This is the module that the component is currently mastered in (from the *effectiveTime* held on the component record). A module is simply a collection of *SNOMED CT components* that are maintained as a unit by a single organization. It is the organization's responsibility to organize the components in each *extension* that it is responsible for into one or more modules, in a way that best fits its business needs.

A module is modeled by a *descendant* of the **|Module| concept** in the metadata *hierarchy*. The **|Module| sub-hierarchy** is organized by a maintaining organization into a number of groups. For example, all modules maintained by *IHTSDO* will be *children* of **|IHTSDO maintained module|**. The **|Module| sub-hierarchy** models modules maintained by each organization and does NOT model module dependencies. Instead, module dependencies are modeled using the **|Module dependency| reference set**.

At the point of release, if any component within a module has changed, then a new row will be added for the module's *concept*, with the *effectiveTime* set to the date of the new release, irrespective of whether the other fields in the module *concept* record itself have changed. The updated **|Module| concept** record identifies that some components within the module have been updated in this release. Where no components within a module have been updated, then a new module record will not be added and the module's *effectiveTime* field will not change from the previous release.

Each *SNOMED CT component* will be in one, and only one module. The module that a component is mastered in may change over time, and when this happens, the component's *moduleId* field will be updated (in the usual way by appending a row for the component).

Each module will be in one and only one *extension*. Modules will not straddle *extensions*. The *extension* that a module resides in is defined by the *SCTID* of the module. A module may not move from one *extension* to another over time. If the components within a module are to be moved to another *extension*, then a new module must be created within the destination *extension* to host the components that are to be transferred.

There may be more than one module in an *extension*.

5.6.2.12.4 Descriptor



The table below holds the "reference set descriptor" active member entries for the "module dependency" reference set.

Table 89: Descriptor Template for Module Dependency Reference Sets

refsetId	referencedComponentId	attributeDescription	attributeType	attributeOrder
Reference set descriptor	Module dependency	Module	Concept type component	0
Reference set descriptor	Module dependency	Source effective time	Time	1
Reference set descriptor	Module dependency	Target effective time	Time	2

5.6.2.12.5 Example Usage



Module version dependencies will be modeled using a *reference set*. A module version may depend on one or more other module versions, and many module versions may have a dependency on a single module version. Cyclic module version dependencies are not allowed. The table below holds example entries for the module dependencies *reference set*:

Table 90: Example rows from Module Dependency Reference Set

refsetId	moduleId	Referenced ComponentId	Source EffectiveTime	Destination EffectiveTime
Module dependency	SNOMED CT Australian extension	SNOMED CT core	T2	T1

refsetId	moduleId	Referenced ComponentId	Source EffectiveTime	Destination EffectiveTime
Module dependency	SNOMED CT Australian Pathology	SNOMED CT Australian extension	T2	T2
Module dependency	SNOMED CT Australian Pathology	SNOMED CT core	T2	T1
Module dependency	SNOMED CT Australian Discharge summary	SNOMED CT Australian extension	T2	T2
Module dependency	SNOMED CT Australian Discharge summary	SNOMED CT core	T2	T1

All dependencies will be described in the *release files*, not just immediate dependencies. It is the responsibility of the organization owning a dependent module to identify all modules on which it depends. Therefore, the |Module dependency| reference set members will be held within the dependent module. This is why the moduleId of the reference set member record will always be the source module.

In the above example, the dependencies describe that the | SNOMED CT Australian Pathology| and the | SNOMED CT Australian Discharge Summary| module versions released at T2 are both dependent on the | SNOMED CT Australian extension | module version in the same release, which is itself dependent on the | SNOMED CT core | module version released at T1.

Any release should consist of a set of module versions that are certified as being compatible. Each release should also identify other existing module versions that are outside the scope of the release, but that the release is dependent on.

As dependencies between module versions are described (not just dependencies between modules), it is possible to describe a dependency from a *current* module in a release to a version of a module in a previous release, if so desired. It is also possible to correct historical dependencies between previous modules if these had previously been stated incorrectly.

5.6.2.13 Description Format Reference Set

5.6.2.13.1 Purpose



The *Description Format* reference set provides format and maximum length information for each *description* type.

5.6.2.13.2 Reference Set Data Structure



The CI (*component - Integer*) reference set format is described below:

Table 91: Description Type Reference Set - Data Structure

Field	Data type	Purpose
id	UUID	A 128 bit unsigned <i>integer</i> , uniquely identifying the <i>reference set member</i> .

Field	Data type	Purpose
<i>effectiveTime</i>	<i>Time</i>	Specifies the inclusive date at which this change becomes effective.
<i>active</i>	<i>Boolean</i>	Specifies whether the member's state was <i>active</i> or <i>inactive</i> from the nominal release date specified by the <i>effectiveTime</i> field.
<i>moduleId</i>	<i>SCTID</i>	Identifies the member version's module. Set to a <i>child</i> of <i> Module </i> within the metadata <i>hierarchy</i> .
<i>refsetId</i>	<i>SCTID</i>	Set to the <i> Description format reference set concept</i> in the metadata <i>hierarchy</i> .
<i>referencedComponentId</i>	<i>SCTID</i>	A reference to a <i>child</i> of <i> Description type </i> in the metadata <i>hierarchy</i>
<i>descriptionFormat</i>	<i>SCTID</i>	A reference to a <i>child</i> of <i> Description format reference set attribute concept</i> in the metadata <i>hierarchy</i> .
<i>descriptionLength</i>	<i>Integer</i>	The maximum length in bytes for <i>descriptions</i> of this <i>description type</i> .

5.6.2.13.3 Metadata



The following metadata supports the *description format reference set*.

- 900000000000454005 | Foundation metadata concept |
 - 900000000000455006 | Reference set |
 - 900000000000538005 | Description format |

Figure 54: Description Format Reference Sets in the Metadata Hierarchy

5.6.2.13.4 Descriptor



The table below holds the Descriptor for the *Description Format reference set*.

Table 92: Descriptor Template for Description Type Reference Sets

refsetId	referencedComponentId	attributeDescription	attributeType	attributeOrder
<i> Reference set descriptor </i>	<i> Description format </i>	<i> Description type </i>	<i> Concept type component </i>	0
<i> Reference set descriptor </i>	<i> Description format </i>	<i> Description format </i>	<i> Concept type component </i>	1

refsetId	referencedComponentId	attributeDescription	attributeType	attributeOrder
Reference set descriptor	Description format	Description length	Unsigned integer	2

5.6.2.13.5 Example Usage



This example holds the entries for the *Description Format* reference set.

Table 93: Example rows from Description Format Reference Set

refsetId	referencedComponentId	descriptionFormat	descriptionLength
Description format	Fully Specified Name	Limited HTML	1024
Description format	Synonym	Limited HTML	1024
Description format	Definition	Limited HTML	1024
Description format	Purpose	Limited HTML	1024

Chapter

6

6 Concept Model Guide



This part of the guide explains the *SNOMED CT Concept Model*. This is the model used to specify logical definitions of *SNOMED CT concepts*. It is based on a combination of formal logic and a set of editorial rules that determined the permitted sets of *attributes* and values that may applied to particular types of *concepts*.

6.1 Essential Features of the Concept Model



This section describes key features of the *Concept Model* that underpin the definitions of all *SNOMED CT concepts*.

6.1.1 Root and top-level Concepts



6.1.1.1 The Root Concept



The *Concepts Table* includes a special *concept* referred to as the *Root Concept*. It is the "root" of the main hierarchy that contain all the *Concepts* in *SNOMED CT*.

All other *Concepts* are descended from this "root" *concept* via at least one series of *Relationships* of the *Relationship Type* | is a | (i.e. all other *Concepts* are regarded as subclasses of this *Concept*).

The *Root Concept* Code is 138875005 and is named | SNOMED CT Concept |.

6.1.1.1.1 Features of the root Concept



All other *SNOMED CT Concepts* are *subtypes* of the root *concept*.

Unlike other *SNOMED CT Concepts*, the root *concept* is not a *subtype* of any other *concept*.

6.1.1.1.2 Release information in the root Concept



The root *Concept* has a *current Synonym* that contains information about the release. The *Synonyms*, representing earlier releases, are distributed as *Inactive Descriptions*. The release information is represented in the *term* text of the *Synonym* as indicated in [Table 94](#).

Table 94: Representation of release information in the root Concept

Example	<i>SNOMED Clinical Terms</i> version: 20020131 [R] (first release)	
Stylized form	<i>SNOMED Clinical Terms</i> version: yyyyymmdd [status] (description)	
	yyyyymmdd	The release date in ISO format.
	Status	R (release), D (developmental) or E (evaluation).
	Description	An optional free text <i>description</i> of the release.

6.1.1.2 Top-level Concepts



Concepts that are directly related to the *Root Concept* by a single *Relationship of the Relationship Type* | are referred to as "Top Level Concepts". All other concepts are descended from at least one Top Level Concept via at least one series of Relationships of the Relationship Type | is a | (i.e. all other concepts represent subclasses of the meaning of at least one Top Level Concept).

Many Top-level Concepts are intended to represent things outside of SNOMED CT (including processes, events, and material entities) in the real world. These include:

Table 95: Top Level Concepts

<ul style="list-style-type: none"> • Clinical finding • Procedure • Observable entity • Body structure • Organism • Substance • Pharmaceutical / biologic product • Specimen • Special concept • SNOMED CT Model Component 	<ul style="list-style-type: none"> • Physical force • Event • Environment or geographical location • Social context • Situation with explicit context • Staging and scales • Physical object • Qualifier value • Record artifact
--	---

6.1.1.2.1 Representation of top-level Concepts



Awareness of the top-level Concepts is likely to be particularly important when developing technical implementations.

A top-level Concept can be identified by the fact that it has a single *subtype relationship* referring to the *Root Concept*. However, to minimize processing requirements the top-level Concepts have designated *Concept Identifiers* that are documented in this guide as *Important Concept Identifiers*.

6.1.1.3 Top Level Metadata Concepts



Metadata codes represent structural information about the terminology itself. The Top Level Metadata Concepts represent broad groups of metadata.

Table 96: Top Level Metadata

<ul style="list-style-type: none"> • Core metadata concept • Foundation metadata concept • Linkage concept • Namespace concept
--

6.1.2 Subtype Relationships



6.1.2.1 Role of subtype Relationships



Subtype Relationships provide the main semantic *hierarchy* that relates Concepts to one another.

All Active Concepts, except the root Concept, have *subtype Relationships* with one or more Concepts. Each of these Relationships indicates that a Concept is a *subtype* of another Concept.

6.1.2.2 Representation of Subtype Relationships



Subtype Relationships are expressed in the same way as all other *SNOMED CT Relationships*.

They are identifiable by their *RelationshipType*, which refers to a *Concept* with the *Fully Specified Name* | is a |.

The *subtype Relationship Concept* has a designated *Concept Identifier*, which is documented in this guide as an *Important Concept Identifier*.

6.1.2.3 Subtype Relationships and the Subtype Hierarchy



Subtype Relationships represent the *subtype hierarchy* of *SNOMED CT*. This is illustrated here using a small sample set of *concepts* and *Relationships* listed in *Table 97*.⁸

Table 97: Subtype Relationships Example

Source	Relationship Type	Destination
bacterial pneumonia	is a	infective pneumonia
bacterial pneumonia	is a	bacterial infectious disease
infective pneumonia	is a	infectious disease
infective pneumonia	is a	pneumonia
pneumonia	is a	disease of lung
disease of lung	is a	disease of respiratory system
disease of respiratory system	is a	disease
bacterial infectious disease	is a	infectious disease
infectious disease	is a	disease
disease	is a	SNOMED CT Concept

Only the most proximate | is a | relationships are represented in the distribution files. These *Relationships* are shown by the blue lines in *Figure 55*. However, a *Concept* is a *subtype* of any *concept* to which it has a direct or indirect | is a | *Relationship*.

- Thus the *Concept* | bacterial pneumonia | is a *subtype* of all the other *concepts* shown in the diagram.

Example:

| Bacterial pneumonia | is a *subtype* of | pneumonia | because it is a *subtype* of | infective pneumonia | which is a *subtype* of | pneumonia |

⁸ Only a small sample of concepts and relationships have been included to produce a simple illustration. Some concept have been omitted and direct relationships have been included where in the release data the relationships pass via additional intermediate concepts.

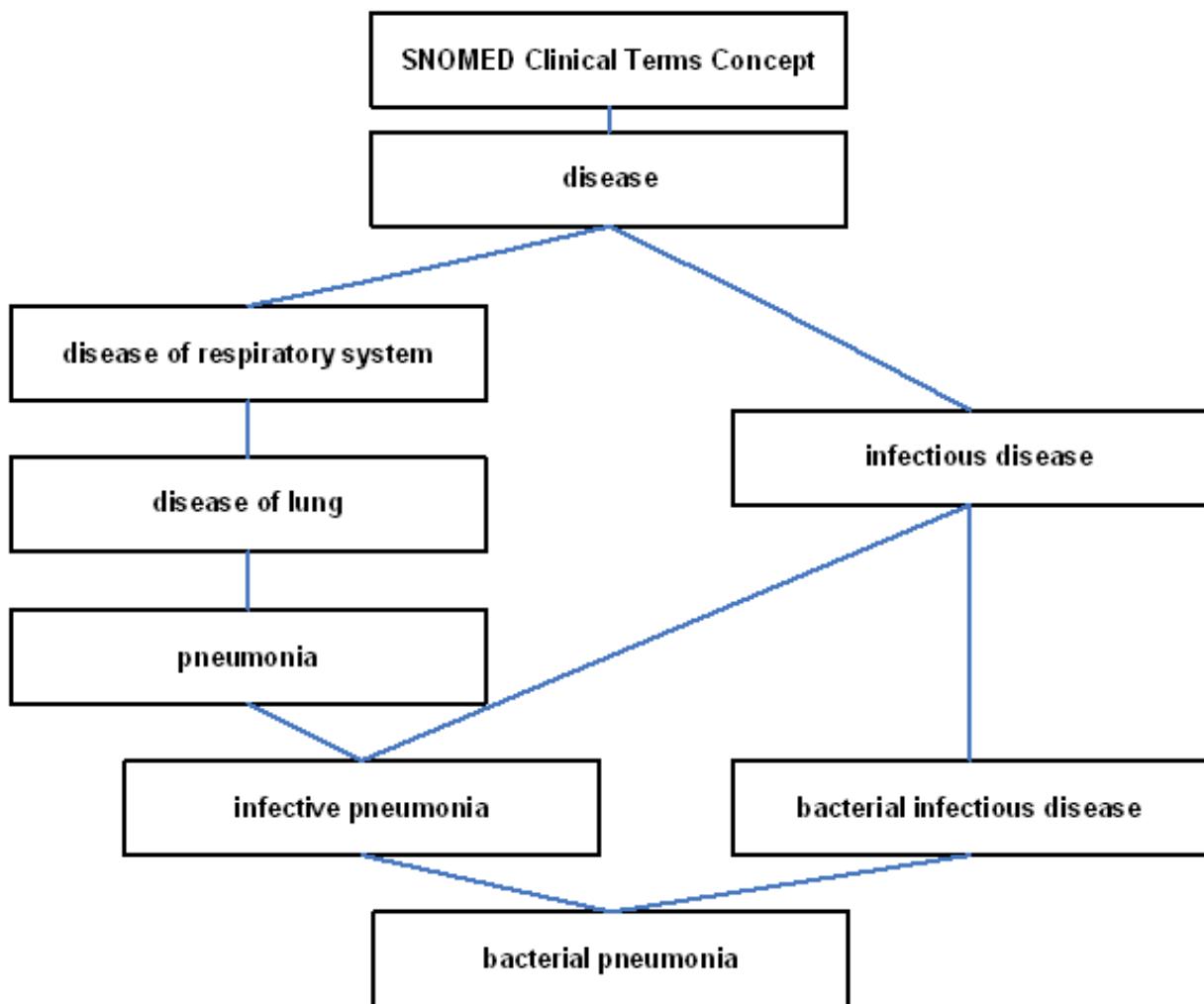


Figure 55: Graphical view of the of the Supertypes of | bacterial pneumonia |

The number of links in the chain of | is a | *Relationships* between two *Concepts* does not alter the logical meaning of the *relationship* between them. The number of | is a | *Relationships* between two *Concepts* may change between releases of *SNOMED CT* as a result of the addition of an intermediate *Concept*. This does not alter the semantic *relationship* between them.

Some technical implementation issues are affected by whether a pair of *Concepts* is linked by a single *subtype Relationship* or by a sequence of several *subtype Relationships*. In this guide, the following *terms* are used where this distinction is technically significant:

A given *Concept* (*Concept-x*) may have:

- *Subtype children* - *Concepts* with a *subtype Relationship* referring to *Concept-x*:
 - | bacterial pneumonia | is a *subtype child* of:
 - | bacterial infectious disease |
 - | infective pneumonia |
- *Supertype parents* - *Concepts* referred to by a *subtype Relationship* from *Concept-x*:
 - | infectious disease | is a *supertype parent* of:
 - | bacterial infectious disease |
 - | infective pneumonia |

- *Subtype descendants* - Concepts with *subtype Relationships* that refer to other Concepts that are either child or subtype descendants of Concept-x:
 - | bacterial pneumonia | is a *subtype descendant* of:
 - All other Concepts shown in the example.
- *Supertype ancestors* - Concepts referred to by *subtype Relationships* from other Concepts that are either parent or supertype ancestors of Concept-x:
 - | disease | is an *supertype ancestor* of:
 - All other Concepts shown in the example, except for | SNOMED CT Concept |.
- | bacterial pneumonia |
 - | bacterial pneumonia | | is a | | infective pneumonia |
 - | infective pneumonia | | is a | | infectious disease |
 - | infectious disease | | is a | | disease |
 - | disease | | is a | | SNOMED CT Concept |
 - | infective pneumonia | | is a | | pneumonia |
 - | pneumonia | | is a | | disease of lung |
 - | disease of lung | | is a | | disease of respiratory system |
 - | disease of respiratory system | | is a | | disease |
 - | disease | | is a | | SNOMED CT Concept |
 - | bacterial pneumonia | | is a | | bacterial infectious disease |
 - | bacterial infectious disease | | is a | | infectious disease |
 - | infectious disease | | is a | | disease |
 - | disease | | is a | | SNOMED CT Concept |

Figure 56: Inverted hierarchical view of the Supertypes of / bacterial pneumonia /

6.1.3 Defining characteristics



6.1.3.1 Role of defining characteristics



Subtype relationships contribute the hierarchical type based aspect of a *Concept* definition. This is augmented by *defining characteristics* that represent the values of a range of relevant attributes. Depending on the nature of the *concept* these may include including etiology, topography, method, etc.

The range of attributes applicable depends on the type of *Concept*. For example, a procedure may have a method, and a disorder may have an etiology, but a procedure cannot have an etiology, and disorder cannot have a method.

Defining characteristics using a particular attribute will be applied consistently to all *Concepts* to which it is relevant. Note that this design principle may not be fully realized for all attributes in each release.

6.1.3.2 Representation of defining characteristics



Defining characteristics are represented as *Relationships*. The fields are used as follows:

- *SourceId* refers to the *Concept* to which a *defining characteristic* applies;
- *TypeId* indicates the nature of the defining attribute;
- *DestinationId* refers to the *Concept* that represents the value of that attribute.

In each release the supported *defining characteristics* for every *Concept* are distributed in the *Relationships Table*. The supported *defining characteristics* are descendants of the concept 410662002 | concept model attribute |. The list of supported defining attributes is provided in [Defining Attributes by Hierarchy and Domain](#)⁹.

Table 98: Defining characteristics applied to an example concept

Disease		
	is a SNOMED CT Concept	
	Primitive	Not all SNOMED CT Concepts are diseases. No <i>defining characteristics</i> are included to specify what makes something a disease.
infectious disease		
	is a disease	
	causative agent infectious agent	
	Primitive	Not all diseases with causative agent infectious agent are bacterial infectious disease . For example, rheumatic heart disease has causative agent streptococcus but is not an infectious disease .
bacterial infectious disease		
	is a infectious disease	
	causative agent bacteria	
	fully defined	All infectious diseases with causative agent bacteria are Bacterial infectious disease
disease of respiratory system		
	is a disease	
	finding site respiratory system structure	

⁹ Note that the Relationships shown in the table and diagram are not the definitive released Relationships of these Concepts. They have been simplified to illustrate particular points in the text.

	<i>fully defined</i>	All diseases with finding site respiratory system structure are Disorder of respiratory system .
disease of lung		
		is a disease of respiratory system
		<i>finding site</i> lung structure
	<i>fully defined</i>	All diseases of respiratory system with finding site lung are Disorder of lung .
Pneumonia		
		is a disease of lung
		<i>finding site</i> lung structure
	<i>Primitive</i>	Not all diseases of lung are pneumonia . No additional characteristics specify what attributes are needed to specify pneumonia
infective pneumonia		
		is a infectious disease
		is a pneumonia
		causative agent infectious agent
		<i>finding site</i> lung structure
	<i>fully defined</i>	All pneumonias with causative agent infectious agent are infective pneumonia .
bacterial pneumonia		
		is a bacterial infectious disease
		is a infective pneumonia
		causative agent bacteria
		<i>finding site</i> lung structure
	<i>fully defined</i>	All pneumonias with causative agent bacteria are bacterial pneumonia .

6.1.4 Qualifiers and refinement



6.1.4.1 Qualifiers and refinable definitions



A *qualifying characteristic* is an attribute that may have one of several possible values for a particular *Concept*. If a particular *qualifier* is applied to a *Concept*, the resulting *expression* represents a more tightly defined *subtype* of that *Concept*.

Example: It might be possible to qualify a disorder such as | bacterial pneumonia | according to its clinical course (| acute | or | chronic |) or severity ("mild," "moderate" or | severe |). With appropriate *qualifiers*, "injury of skin of the left side of face" could then be represented even if a single *Concept Identifier* cannot express this.

A similar tightening of the definition of a *Concept* can be achieved by allowing one or more of the *defining characteristics* associated with a *Concept* to be refined. A *defining characteristic* is refined by an *expression* that applies a specified *subtype* of the value stated in the definition.

Example: | Fracture of bone | could be refined by qualifying it with the finding site "tibia" to represent the *Concept* | Fracture of tibia |.

6.1.5 Primitive and fully-defined Concepts



A *Concept* is considered to be *fully defined* if its *defining characteristics* are sufficient to define it relative to its immediate supertype(s). A *Concept* which is not *fully defined* is *Primitive* and this is indicated by the value of the *definitionStatusId* field.

Example: | Pneumonia | is a lung disease but unless *defining characteristics* are specified that effectively distinguish | pneumonia | from other lung diseases then it is regarded as a *primitive Concept*.

If a *Concept* is *primitive* then the *defining characteristics* for that *Concept* are incomplete. It is not possible to automatically compute that a *Concept* represented as a *postcoordinated combination* of several *Concepts* is or is not a *subtype* of a particular *primitive Concept*.

Example: The *Concept* "lung disease" qualified by | causative agent | = | bacteria | may be | pneumonia | but could also be "bronchitis."

In contrast if a *Concept* is *fully defined* it is possible to state that any *Concept* represented as a combination of the same *defining characteristics* is equivalent to or a *subtype* of that *Concept*.

Example: Assume that the *Concept* | bacterial pneumonia | is *fully defined* as | infective pneumonia | with | causative agent | = | bacteria | and that | pneumococcus | is a | bacteria |. It then follows that the post coordinated representation of | pneumococcal pneumonia | as | infective pneumonia | with | causative agent | = | pneumococcus | is computably a *subtype* of | bacterial pneumonia |.

6.1.6 Important Concept Identifiers



Table 99: Root Concept and Subtype Relationship

Id	Preferred Term	Comments
138875005	SNOMED CT Concept	<p>All <i>Active Concepts</i> are <i>subtype descendants</i> of this <i>Root Concept</i>.</p> <p>The <i>Root Concept</i> has a <i>current Synonym</i> representing the release date.</p>

Id	Preferred Term	Comments
116680003	is a	Relates a <i>Concept</i> to its immediate supertype <i>Concepts</i> .

Table 100: Top-Level Concepts

Id	Preferred Term	
123037004	body structure	
404684003	clinical finding	
308916002	environment or geographical location	
272379006	event	
363787002	observable entity	
410607006	organism	
373873005	pharmaceutical / biologic product	
78621006	physical force	
260787004	physical object	
71388002	procedure	
362981000	qualifier value	
419891008	record artefact	
243796009	situation with explicit context	
48176007	social context	
123038009	specimen	
254291000	staging and scales	
105590001	substance	
106237007	linkage concept	In <i>Release Format 2</i> this is a <i>subtype</i> of SNOMED CT Model Component .

Id	Preferred Term	
370115009	special concept	Not used in <i>Release Format 2</i> . Replaced and extended by SNOMED CT Model Component .
90000000000441003	SNOMED CT Model Component	Introduced in <i>Release Format 2</i> .

Table 101: Special Concepts

Id	Preferred Term	Comments
370115009	special concept	A top-level <i>Concept</i> that has as its immediate <i>subtypes</i> a set of <i>Concepts</i> that are used to support the functionality of the terminology rather than to represent real-world <i>Concepts</i> .
362955004	inactive concept	A Special <i>Concept</i> with immediate <i>subtypes</i> that represent each of the possible <i>inactive ConceptStatus</i> values. Each of these has as its immediate <i>subtypes</i> all <i>Inactive Concepts</i> that have that <i>ConceptStatus</i> .
370136006	namespace concept	A Special <i>Concept</i> with immediate <i>subtypes</i> that each represents a <i>SNOMED CT namespace</i> . There is one <i>Namespace Concept</i> for the <i>SNOMED CT core</i> and one additional <i>Namespace Concept</i> for each <i>Extension</i> for which a <i>namespace-Identifier</i> has been allocated.
363743006	navigational concept	A Special <i>Concept</i> that has as its immediate <i>subtypes</i> all <i>active Navigation Concepts</i> .

Table 102: Historical RelationshipType Concepts (Not used in RF2)

Id	Preferred Term	Comment
149016008	MAY BE A	Relates an ambiguous <i>Concept</i> to the <i>Concepts</i> that represent each of its possible meanings.
384598002	MOVED FROM	Relates to a <i>Concept</i> in another <i>namespace</i> that is replaced by this <i>Concept</i> . The target <i>Concept</i> remains in the original <i>namespace</i> with the <i>inactive ConceptStatus</i> "Moved Elsewhere" or, for a limited period during handover, the <i>active status</i> "Pending Move." The target <i>Concept</i> has a Moved to Relationship pointing to the <i>namespace</i> in which the <i>Concept</i> is now supported.

Id	Preferred Term	Comment
370125004	MOVED TO	<p>Relates to the <i>Namespace Concept</i> for a different <i>namespace</i> in which this <i>Concept</i> is now maintained.</p> <p>The source <i>Concept</i> must have the <i>ConceptStatus</i> "Moved elsewhere" or "Pending move."</p> <p>The target <i>namespace</i> contains the <i>active</i> version of this <i>Concept</i> with a Moved from Relationship to the <i>Concept replaced concept</i>.</p>
370124000	REPLACED BY	Relates an erroneous <i>Concept</i> to a corrected <i>Concept</i> that replaces it.
168666000	SAME AS	Relates a duplicate <i>Concept</i> with an <i>Active Concept</i> that has the same meaning.
159083000	WAS A	Relates a <i>Concept</i> to an <i>Inactive Concept</i> that was formerly considered to be one of its supertypes.

Table 103: Valid Relationship Type Concepts - Defining Characteristics

Id	Preferred Term	Comment
246061005	attribute	
410662002	concept model attribute	
260507000	access	
246090004	associated finding	
116676008	associated morphology	
363589002	associated procedure	
47429007	associated with	
255234002	after	<i>Subtype of associated with .</i>
246075003	causative agent	<i>Subtype of associated with .</i>
42752001	due to	<i>Subtype of associated with .</i>
263502005	clinical course	
246093002	component	

<i>Id</i>	<i>Preferred Term</i>	<i>Comment</i>
363701004	direct substance	
246456000	episodicity	
408729009	finding context	
419066007	finding informer	
418775008	finding method	
363698007	finding site	
127489000	has active ingredient	
363705008	has definitional manifestation	
411116001	has dose form	
363702006	has focus	
363703001	has intent	
363713009	has interpretation	
116686009	has specimen	
363714003	interprets	
272741003	laterality	
370129005	measurement method	
260686004	method	
246454002	occurrence	
123005000	part of	
370135005	pathological process	
260870009	priority	
408730004	procedure context	

<i>Id</i>	<i>Preferred Term</i>	<i>Comment</i>
405815000	procedure device	
363699004	direct device	<i>Subtype of procedure device .</i>
363710007	indirect device	<i>Subtype of procedure device .</i>
424226004	using device	<i>Subtype of procedure device .</i>
425391005	using access device	<i>Subtype of using access device .</i>
405816004	procedure morphology	
363700003	direct morphology	<i>Subtype of procedure morphology .</i>
363709002	indirect morphology	<i>Subtype of procedure morphology .</i>
363704007	procedure site	
405813007	procedure site - Direct	<i>Subtype of procedure site .</i>
405814001	procedure site - Indirect	<i>Subtype of procedure site .</i>
370130000	property	
370131001	recipient category	
246513007	revision status	
410675002	route of administration	
370132008	scale type	
246112005	severity	
118171006	specimen procedure	
118170007	specimen source identity	
118168003	specimen source morphology	
118169006	specimen source topography	
370133003	specimen substance	

Id	Preferred Term	Comment
131195008	subject of information	
408732007	subject relationship context	
424876005	surgical approach	
408731000	temporal context	
370134009	time aspect	
424244007	using energy	
424361007	using substance	

6.2 Concept Model Specification



This section specifies the main hierarchies of the *SNOMED CT Concept Model* and the attributes and values used to define *concepts* in particular hierarchies.

6.2.1 Hierarchies



SNOMED CT concepts are organized into hierarchies. There is one special *concept* referred to as the | Root Concept Code | . It represents the "root" of the hierarchy that contains all *Concepts* in *SNOMED CT*. The root named "*SNOMED CT Concept*" subsumes (is the supertype of) the top-level *concepts* (hierarchies parents) and all the *concepts* beneath them (their *subtypes*). As the hierarchies are descended, the *concepts* within them become increasingly specific (or granular). A brief description of the content in each *hierarchy* is given below.

Subtype (or "child") *concepts* are the *descendant concepts* of *Supertype* (or "parent") *concepts*.

👉 **Example:** | Streptococcal arthritis (disorder) | is a *subtype* of | Bacterial arthritis (disorder) |.

Supertype concepts are the *ancestor concepts* of *Subtype concepts*.

👉 **Example:** | Bacterial arthritis (disorder) | is a *supertype* of | Streptococcal arthritis (disorder) |.

6.2.1.1 Summary of Top Level Hierarchies

6.2.1.1.1 Top Level Concepts



Table 104: Top Level Concepts

<ul style="list-style-type: none"> • Clinical finding • Procedure • Observable entity • Body structure • Organism • Substance • Pharmaceutical / biologic product • Specimen • Special concept • SNOMED CT Model Component 	<ul style="list-style-type: none"> • Physical force • Event • Environment or geographical location • Social context • Situation with explicit context • Staging and scales • Physical object • Qualifier value • Record artifact
--	---

6.2.1.1.2 Top Level Metadata



Table 105: Top Level Metadata

<ul style="list-style-type: none"> • Core metadata concept • Foundation metadata concept • Linkage concept • Namespace concept
--

6.2.1.2 Clinical finding



Concepts in this *hierarchy* represent the result of a clinical observation, assessment or judgment, and include both normal and abnormal clinical states.

Examples of Clinical finding concepts:

- | Clear sputum (finding) | ;
- | Normal breath sounds (finding) | ;
- | Poor posture (finding) | .

The | Clinical finding | *hierarchy* contains the *sub-hierarchy* of | Disease |. Concepts that are *descendants* of | Disease | (or disorders) are always and necessarily abnormal clinical states. The *subtype polyhierarchy* allows diseases to be *subtypes* of other disorders as well as *subtypes* of findings.

Examples of Disease concepts:

- | Tuberculosis (disorder) | ;
- | non-Hodgkin's lymphoma (disorder) | .

 **Note:** See also [Attributes used to define Clinical Finding concepts](#).

6.2.1.3 Procedure



| Procedure | *concepts* represent activities performed in the provision of health care. This *hierarchy* represents a broad variety of activities, including but not limited to, invasive procedures (e.g. | Excision of intracranial artery (procedure) |), administration of medicines (e.g. | Pertussis vaccination (procedure) |),

imaging procedures (e.g. | Ultrasonography of breast (procedure) |), education procedures (e.g. | Low salt diet education (procedure) |), and administrative procedures (e.g. | Medical records transfer (procedure) |).

Examples of Procedure concepts:

- | Removal of urethral catheter (procedure) | ;
- | Intravenous steroid injection (procedure) | ;
- | Irrigation of oral wound (procedure) | ;
- | Appendectomy (procedure) | .

 **Note:** See also [Attributes used to define Procedure concepts](#).

6.2.1.4 Situation with explicit context



Concepts in the | Situation with explicit context | hierarchy (given the appropriate record structure) can be used in a clinical record to represent:

- Conditions and procedures that have not yet occurred (e.g. | Endoscopy arranged (situation) |);
- Conditions and procedures that refer to someone other than the patient (e.g. | Family history: Diabetes mellitus (situation) |, | Discussed with next of kin (situation) |);
- Conditions and procedures that have occurred at some time prior to the time of the *current* entry in the record (e.g. | History of - aortic aneurysm (situation) |, | History of - splenectomy (situation) |).

In each of these examples, clinical context is specified. The second example, in which someone other than the patient is the focus of the *concept*, could be represented in an application or record structure by combining a header *term* Family history with the value Diabetes. The specific context (in this case, family history) would be represented using the record structure. In this case, the *precoordinated* context-dependent concept | Family history: Diabetes mellitus (situation) | would not be used because the information model has already captured the family history aspect of the diabetes.

Concepts in the | Procedure | and | Clinical finding | hierarchy have a default context of the following:

- The procedure **has actually occurred** (versus being planned or canceled) or the finding is actually present (versus being ruled out, or considered);
- The procedure or finding being recorded **refers to the patient of record** (versus, for example, a family member);
- The procedure or finding **is occurring now or at a specified time** (versus some time in the past).

In addition to using the record structure to represent context, there is sometimes a need to override these defaults and specify a particular context using the formal logic of the terminology. For that reason, SNOMED CT has developed a context model to allow users and/or implementers to specify context using the terminology, without depending on a particular record structure. The | Situation with explicit context | hierarchy and various attributes assigned to *concepts* in this hierarchy accomplish this.

Examples of Situation with explicit context concepts:

- | Family history: Myocardial infarction (situation) | ;
- | No family history of stroke (situation) | ;
- | Nasal discharge present (situation) | ;
- | Suspected epilepsy (situation) | .

 **Note:** See also [Attributes used to define Situation with Explicit Context concepts](#).

6.2.1.5 Observable entity



Concepts in this hierarchy can be thought of as representing a question or procedure which can produce an answer or a result. For instance, | Left ventricular end-diastolic pressure (observable entity) | could be interpreted as the question, "What is the left ventricular end diastolic pressure?" or "What is the measured left ventricular end-diastolic pressure?"

Observables are entities that could be used to code elements on a checklist or any element where a value can be assigned. | Color of nail (observable entity) | is an observable. | Gray nails (finding) | is a finding.

One use for | Observable entity | in a clinical record is to code headers on a template. For example, | Gender (observable entity) | could be used to code a section of a template titled “Gender” where the user would choose “male” or “female”. “Female gender” would then constitute a finding.

 **Note:** See also [Attributes used to define Evaluation Procedure concepts](#).

6.2.1.6 Body structure



| Body structure | *concepts* include normal as well as abnormal anatomical structures. Normal anatomical structures can be used to specify the body site involved by a disease or procedure.

Examples of Body structure concepts:

- | Mitral valve structure (body structure) | ;
- | Uterine structure (body structure) | .

Morphologic alterations from normal body structures are represented in the *sub-hierarchy* | Body structure, altered from its original anatomical structure (morphologic abnormality) |.

Examples of Body Structure, altered from its original anatomical structure concepts:

- | Adenosarcoma (morphologic abnormality) | ;
- | Polyp (morphologic abnormality) | .

 **Note:** See also [Attributes used to define Body structure concepts](#).

6.2.1.7 Organism



This *hierarchy* includes organisms of significance in human and animal medicine. Organisms are also used in modeling the causes of diseases in *SNOMED CT*. They are important for public health reporting of the causes of notifiable conditions and for use in evidence-based infectious disease protocols in clinical decision support systems. Sub-hierarchies of organism include, but are not limited to: | Animal (organism) |, | Microorganism (organism) |, | Kingdom Plantae (organism) |.

Examples of Organism concepts:

- | Streptococcus pyogenes (organism) | ;
- | Texon cattle breed (organism) | ;
- | Bacillus anthracis (organism) | ;
- | Lichen (plant) (organism) | .

6.2.1.8 Substance



The | Substance | *hierarchy* contains *concepts* that can be used for recording *active* chemical constituents of drug products, food and chemical allergens, adverse reactions, toxicity or poisoning information, and physicians and nursing *orders*. *Concepts* from this *hierarchy* represent general substances and chemical constituents of | Pharmaceutical / biologic product (product) | which are in a separate *hierarchy*. However, sub-hierarchies of | Substance | also include but are not limited to: | Body substance (substance) | (*concepts* to represent body substances); | Dietary substance (substance) |; | Diagnostic substance (substance) |.

Examples of Substance concepts:

- | Insulin (substance) | ;
- | Methane (substance) | ;
- | Chromatin (substance) | ;
- | Dental porcelain material (substance) | ;
- | Albumin (substance) | ;

- | Endorphin (substance) | ;
- | Acetaminophen (substance) | .

6.2.1.9 Pharmaceutical/biologic product



The | Pharmaceutical / biologic product | *hierarchy* is separate from the | Substance | *hierarchy*.

This *hierarchy* was introduced as a top-level *hierarchy* in order to clearly distinguish drug products (products) from their chemical constituents (substances).

It contains *concepts* that represent the multiple levels of granularity required to support a variety of uses cases such as computerized provider order entry (CPOE), e-prescribing, decision support and formulary management. The levels of drug products represented in the *International Release* include Virtual Medicinal Product (VMP), Virtual Therapeutic Moiety (VTM), and Product Category. Additionally, US and UK drug extensions have been developed, which represent Actual Medicinal Products (AMPs).

 **Note:** See also *Attributes used to define Pharmaceutical/Biologic Product concepts*.

6.2.1.9.1 Virtual Medicinal Product (VMP)



The most granular level is the Virtual Medicinal Product (VMP). The VMP is a representation at the level of generality that would appear on a physician's prescription. The product name, strength, and dose form are all represented in the *Fully Specified Name*. This level can be used to support providers with drug ordering in CPOE and e-prescribing use cases.

 **Example:** | Diazepam 5mg tablet (product) |

- (Name, Strength, Dose form).

6.2.1.9.2 Virtual Therapeutic Moiety (VTM)



The Virtual Therapeutic Moiety (VTM) level represents a more general level of granularity than the VMP level. VTMs include the product name but not formulation, dose or strength in the *Fully Specified Name*. The HAS ACTIVE INGREDIENT attribute (which relates the product to the | Substance | it contains) can be assigned to this level or to any of the *subtypes* of this level.

 **Example:** | Diazepam (product) |

All Virtual Medicinal Products (VMP) have a direct link to the Virtual Therapeutic Moiety (VTM) via an | *is a* | *relationship*.

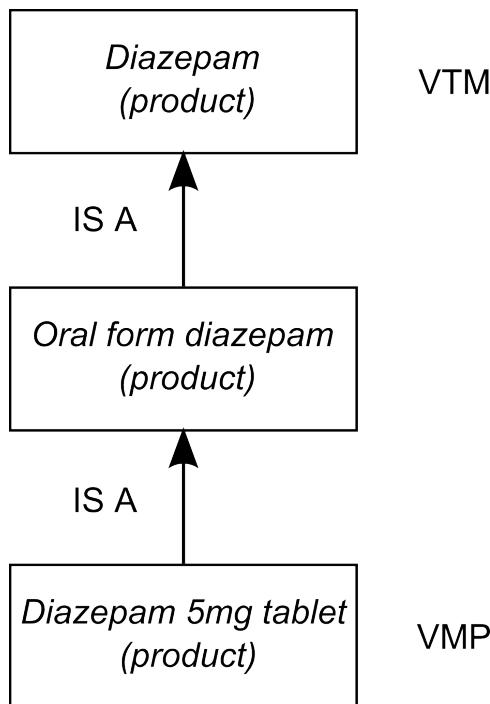


Figure 57: Example

There are additional levels in the | Pharmaceutical / biologic product | *hierarchy* that provide structure and organization . For example, some *subtypes* of VTM contain only Dose form information and not Strength.

👉 **Example:** Concept with granularity between that of a VTM and VMP:

- | Parenteral form epinephrine (product) |:
 - (Dose form, Name).

6.2.1.9.3 Product category



A Product category concept supports a group of | Pharmaceutical / biologic product | related by their functionality mechanism of action or therapeutic use. | Product category | concepts typically describe common drug categories used in prescribing.

Examples of Product category concepts:

- | Sex hormone product (product) | ;
- | Mineralocorticoid preparation (product) | ;
- | beta-Blocking agent (product) | ;
- | Tissue plasminogen activator preparation (product) | .

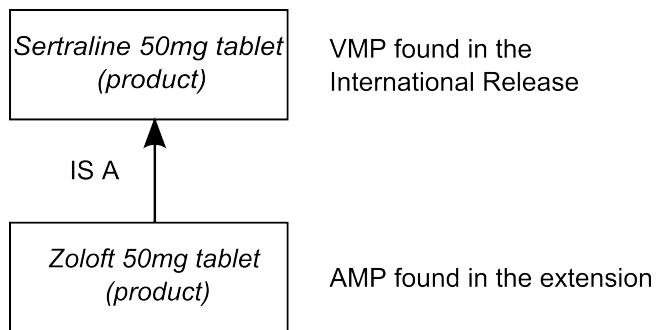
6.2.1.9.4 Actual Medicinal Products (AMPs)



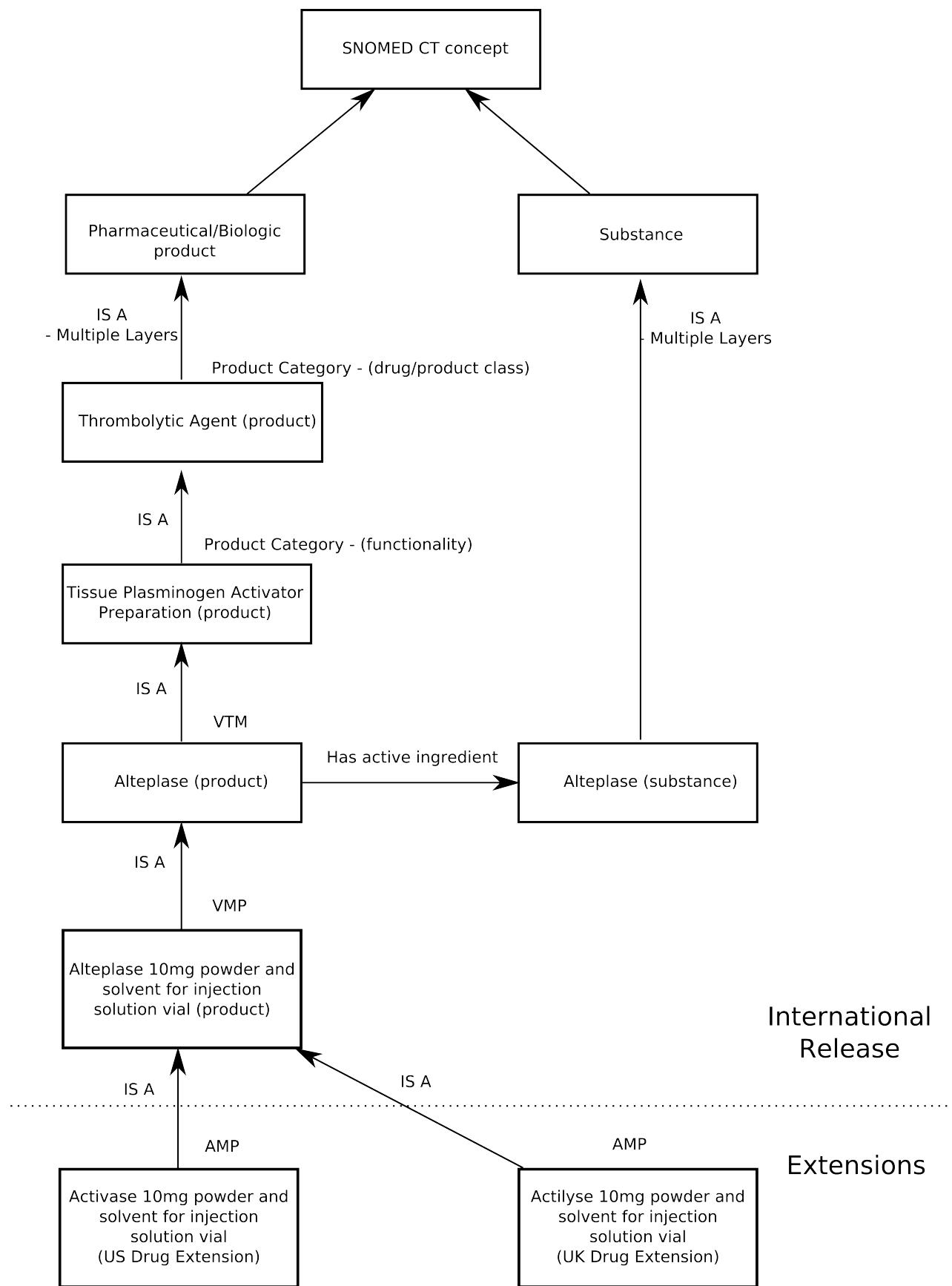
Actual Medicinal Products can be represented in *extensions*. The AMP represents the single unit dose of a medicinal product that is (or has been) made or marketed by a specific manufacturer (trademarked brand name pharmaceutical products). Its *description* requires product name, strength, dosage form, flavor (where applicable) and manufacturer, but it does not include explicit information about packaging.

Because AMP concepts contain brand and country-specific information, they are not represented within the *International Release* of SNOMED CT, but may instead exist within an identified domain extension (contact your IHTSDO National Release Center Center for further information). Actual Medicinal Products in an extension have a direct link to their virtual equivalent in the *International Release* via the | is a | relationship.

Example:



All *concepts* in the | Pharmaceutical / biologic product | *hierarchy* have a FSN tag of "(product)" regardless of their level of granularity.

**Figure 58: Pharmaceutical/Biologic Product hierarchy structure**



6.2.1.10 Specimen

The | Specimen | *hierarchy* contains *concepts* representing entities that are obtained (usually from a patient) for examination or analysis. | Specimen | *concepts* can be defined by attributes which specify: the normal or abnormal body structure from which they are obtained; the procedure used to collect the specimen; the source from which it was collected; and the substance of which it is comprised.

Examples of Specimen concepts:

- | Specimen from prostate obtained by needle biopsy (specimen) | ;
- | Urine specimen obtained by clean catch procedure (specimen) | ;
- | Calculus specimen (specimen) | ;
- | Cerebroventricular fluid cytologic material (specimen) | .

👉 Note: See also [Attributes used to define Specimen concepts](#).



6.2.1.11 Physical object

Concepts in the | Physical object | *hierarchy* include natural and man-made objects. One use for these *concepts* is modeling procedures that use devices (e.g.catheterization).

Examples of Physical object concepts:

- | Military vehicle (physical object) | ;
- | Implant, device (physical object) | ;
- | Artificial kidney, device (physical object) | ;
- | Latex rubber gloves (physical object) | ;
- | Book (physical object) | ;
- | Pressure support ventilator (physical object) | ;
- | Vena cava filter (physical object) | .

👉 Note: See also [Attributes used to define Physical Object concepts](#).



6.2.1.12 Physical force

The *concepts* in the | Physical force | *hierarchy* are directed primarily at representing physical forces that can play a role as mechanisms of injury.

Examples of Physical force concepts:

- | Spontaneous combustion (physical force) | ;
- | Alternating current (physical force) | ;
- | Friction (physical force) | .



6.2.1.13 Event

The | Event | *hierarchy* includes *concepts* that represent occurrences (excluding procedures and interventions).

Examples of Event concepts:

- | Flood (event) | ;
- | Bioterrorist attack (event) | ;
- | Earthquake (event) | .

👉 Note: See also [Attributes used to define Event concepts](#).

6.2.1.14 Environments and geographic locations



The | Environment or geographical location | *hierarchy* includes types of environments as well as named locations such as countries, states, and regions.

Examples of Environments and geographic locations concepts:

- | Canary islands (geographic location) | ;
- | California (geographic location) | ;
- | Rehabilitation department (environment) | ;
- | Intensive care unit (environment) | .

6.2.1.15 Social context



The | Social context | *hierarchy* contains social conditions and circumstances significant to healthcare. Content includes such areas as family *status*, economic *status*, ethnic and religious heritage, life style, and occupations. These *concepts* represent social aspects affecting patient health and treatment. Some sub-hierarchies of | Social context | and *concepts* typical of those sub-hierarchies are shown in the following examples.

Examples:

- | Ethnic group (ethnic group) |:
 - | Afro-Caribbean (ethnic group) | ;
 - | Estonians (ethnic group) | .
- | Occupation (occupation) |:
 - | Bank clerk (occupation) | ;
 - | Carpenter, general (occupation) | .
- | Person (person) |:
 - | Employer (person) | ;
 - | Boyfriend (person) | ;
 - | Caregiver (person) | .
- | Religion / philosophy (religion/philosophy) |:
 - | Hinduism (religion/philosophy) | ;
 - | Orthodox Christian religion (religion/philosophy) | .
- | Economic status (social concept) |:
 - | Middle class economic status (social concept) | .

6.2.1.16 Staging and scales



This *hierarchy* contains such sub-hierarchies as | Assessment scales (assessment scale) |, which names assessment scales; and | Tumor staging (tumor staging) | , which names tumor staging systems.

Examples of Assessment scales (assessment scale) concepts:

- | Glasgow coma scale (assessment scale) | ;
- | Stanford Binet intelligence scale (assessment scale) | .

Examples of Tumor staging (tumor staging) concepts:

- | International Federation of Gynecology and Obstetrics (FIGO) staging system of gynecological malignancy (tumor staging) | ;

- | Dukes staging system (tumor staging) | .

6.2.1.17 Qualifier value



The | Qualifier value | *hierarchy* contains some of the *concepts* used as values for *SNOMED CT* attributes that are not contained elsewhere in *SNOMED CT*. Such a code may be used as the value of an attribute in a defining *Relationship* in *precoordinated* definitions, and/or as the value of an attribute in a *qualifier* in a *postcoordinated expression*. However, the values for attributes are not limited to this *hierarchy* and are also found in hierarchies other than | Qualifier value |.

For example, the value for the attribute | LATERALITY | in the *concept* shown below is taken from the | Qualifier value | *hierarchy*:

- | Left kidney structure | | LATERALITY | | Left | .

However, the value for the attribute | FINDING SITE | in the *concept* shown below is taken from the | Body structure | *hierarchy*, not the | Qualifier value | *hierarchy*.

- | Pneumonia | | FINDING SITE | | Lung structure | .

Examples of Qualifier value concepts:

- | Unilateral | ;
- | Left | ;
- | Puncture - action | .

6.2.1.18 Special concept



The Top Level *Concept Code* | Special concept | and its subclass codes provide a place for *concept* codes that are no longer *active* in the terminology.

The subclasses of | Special concept | are:

- | Navigational concept | ;
- | Inactive concept | .

6.2.1.18.1 Navigational concept



These *concept* codes are to be used only as nodes in alternative navigation structures.

Navigational *concepts* are distributed as active *concepts*, with an inactive *Is-a Relationship* to the *concept* "Navigational concept". These *concepts* should have no *subtypes*.

6.2.1.18.2 Inactive concept (RF2)



When a *concept* is no longer intended for active use, the *concept* and its *Relationships* are turned inactive. Previously, in RF1, the *concept* was moved into an special "Inactive concept" hierarchy, this is not done anymore in RF2, the *concept* is inactivated "in place", with its last location described in the history of its inactive *Relationships*.

6.2.1.18.3 Namespace concept



The *concept* 370136006 | Namespace concept (namespace concept) | is a *subtype* of | SNOMED CT model component |. Each of its *subtype concepts* has an integer term which is an assigned *Extension namespace identifier*.

6.2.1.19 Record artifact



A | Record artifact | is an entity that is created by a person or persons for the purpose of providing other people with information about events or states of affairs. In general, a record is virtual, that is, it is independent of its particular physical instantiation(s), and consists of its information elements (usually words, phrases and sentences, but also numbers, graphs, and other information elements). | Record artifact | need not be complete reports or complete records. They can be parts of larger | Record artifact |. For example, a complete health record is a | Record artifact | that also may contain other | Record artifact | in the form of

individual documents or reports, which in turn may contain more finely granular | Record artifact | such as sections and even section headers.

6.2.1.20 Core metadata concept



Subtypes of | Core metadata concept | provide structural information required to support International Release data. This supporting information includes sets of enumerated values that apply to attributes of concepts, descriptions and relationships.

6.2.1.21 Foundation metadata concept



Subtypes of the | Foundation metadata concept | provide supporting metadata and structural information for derivative release structures including Reference Sets.

6.2.1.22 Linkage concept



Linkage concept codes are intended to link two or more other codes to each other to express compositional meanings. All concept codes that can be used as a Relationship Type are included under | Linkage concept |. The ones approved for use are the Concept Model Attributes. Implementation guidance is as yet quite limited for the other Linkage concept codes. Use of them should be regarded as non-standard, tentative and experimental, requiring extra care.

| Linkage concept | is a subclass of | SNOMED CT model component |, and the | Linkage concept | hierarchy contains the sub-hierarchies:

- | Link assertion |;
- | Attribute |.

👉 **Note:** In RF1, | Linkage concept | was a top level hierarchy.

6.2.1.22.1 Link assertion



The Link assertion sub-hierarchy enables the use of *SNOMED CT concepts* in *HL7* statements that assert *relationships* between statements. Currently this content supports the *UK NHS Connecting for Health* requirements for encoding of Statement *relationships* for the implementation of *HL7* Version 3 messaging in the *UK realm*.

Examples of Link assertion concepts:

- | Has reason | ;
- | Has explanation | .

6.2.1.22.2 Attribute



Concepts that descend from this sub-hierarchy are used to construct *relationships* between two *SNOMED CT concepts*, since they indicate the *relationship type* between those *concepts*. Some attributes (*relationship types*) can be used to logically define a *concept* (defining attributes). This sub-hierarchy also includes non-defining attributes or attributes that may be useful to model *concept* definitions but which have not yet been used in modeling precoordinated concepts in *SNOMED CT*.

Examples of Defining attributes:

- | is a | .
- | Concept model attribute |:
 - | Laterality | ;
 - | Procedure site | ;
 - | Finding site | ;
 - | Associated morphology | .

Examples of Non-defining attributes:

- | Unapproved attribute | :
- | Relieved by | ;
- | Has assessment | .

6.2.2 Attributes Used in SNOMED CT



This part of the guide provides an overview of the defining attributes used by the *SNOMED CT Concept Model*. Further details are provided in the chapters dedicated to each hierarchy.

6.2.2.1 Introduction



SNOMED CT currently uses over 50 defining attributes to model *concept* definitions.

Each *SNOMED CT* attribute can usually be applied to one *hierarchy* and for a few attributes to more than one *hierarchy*. The *hierarchy* or hierarchies to which an attribute can be applied are referred to as the “domain” of the attribute. Each attribute can be given a limited set of values; this set of values is called the “range” of the attribute.

6.2.2.1.1 Domain



The *Domain* is the *hierarchy* to which a specific attribute can be applied.

The *Domain* of the attribute | ASSOCIATED MORPHOLOGY | is the | Clinical finding | *hierarchy*.

A | Procedure | cannot have an | ASSOCIATED MORPHOLOGY |.

A | Procedure | has a | PROCEDURE MORPHOLOGY |.

6.2.2.1.2 Allowable domains in postcoordinated expressions



The *concept model* provides *constraints* for attributes that are used as defining *relationships*, both in distributed *SNOMED CT* content (so-called *precoordinated definitions*) and in *postcoordinated expressions*, as described in the document *Abstract Logical Models and Representational Forms* (available at www.ihtsdo.org/our-standards/technical-documents/). The domain (or starting *concept*) to which qualifying *relationships* are applied in *postcoordinated expressions* may be more general than the domain of defining *relationships* defined in the *concept model*, as long as the resulting *postcoordinated expression* as a whole satisfies the *concept model constraints*.

For example, the *concept model constraint* for | SURGICAL APPROACH | requires that its domain be | Surgical procedure (procedure) | 387713003. When | SURGICAL APPROACH | is used in a qualifying *relationship* in *postcoordinated expressions*, the starting domain may be a general procedure, if the resulting *expression* satisfies the *concept model constraint*. In other words, when | SURGICAL APPROACH | is added to a general procedure as a qualifying *relationship*, the *postcoordinated expression* should also have a METHOD with a value of | Surgical action (qualifier value) | or one of its *subtypes*, so that the resulting *concept* becomes a *subtype* of | Surgical procedure (procedure) |.

6.2.2.1.3 Range



The *Range* is the set of values allowed for each attribute.

For example, the *Range* for | ASSOCIATED MORPHOLOGY | is | Morphologically abnormal structure (morphologic abnormality) | and its *descendants*, and the *Range* for | FINDING SITE | is | Anatomical or acquired body structure (body structure) | and its *descendants* in the | Body structure | *hierarchy*.

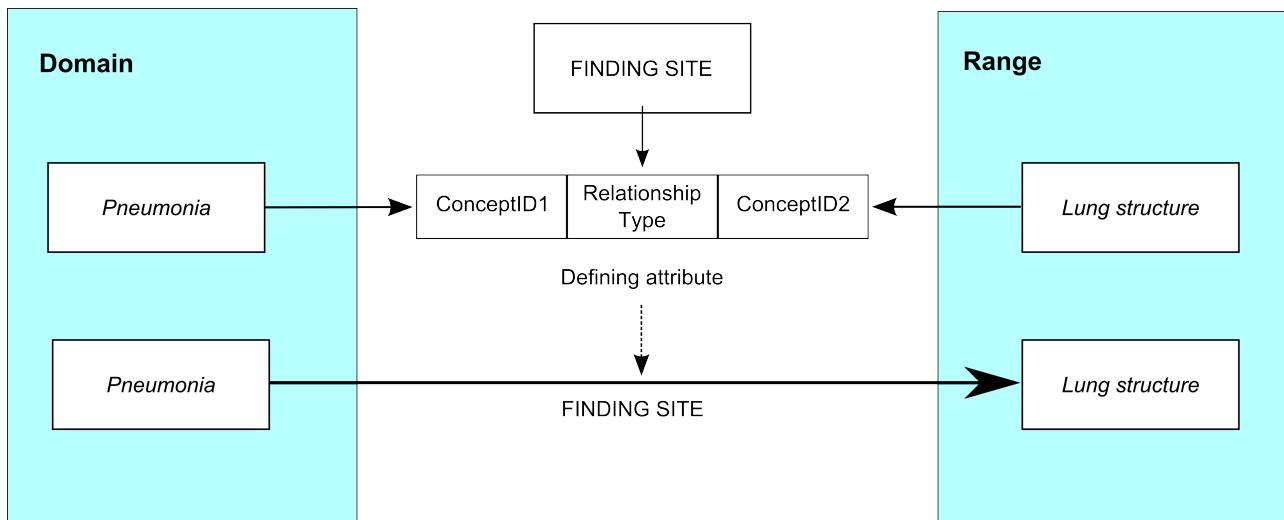


Figure 59: Example Pneumonia FINDING SITE Lung structure

The *Domain* for the | FINDING SITE | attribute is the | Clinical finding | *hierarchy*. In the above example, the attribute | FINDING SITE | has the value | Lung structure (body structure) |. | Lung structure (body structure) | is found in the | Anatomical structure (body structure) | subhierarchy which is in the allowed range for | FINDING SITE |.

Defining attributes in *SNOMED CT* are assigned to the hierarchies where retrieval of clinical data is most useful and relevant (e.g. | Procedure |, | Clinical finding |, | Pharmaceutical / biologic product |, | Situation with explicit context |, | Event |, | Specimen | and | Physical object |). In addition, | LATERALITY | is a defining attribute applied to | Body structure | *concepts*. Other hierarchies, such as | Social context |, | Substance |, | Organism |, and | Observable entity |, are not assigned attributes and instead are considered supporting hierarchies. *Concepts* from the supporting hierarchies can serve as the *attribute values* for the *concept definitions* of the main hierarchies.

This section describes the approved attributes used in *SNOMED CT*. There are many other attributes in *SNOMED CT*, *subtypes* of | Unapproved attribute (attribute) |, which have not yet been evaluated thoroughly and approved for use.

6.2.2.2 Attribute Hierarchies in *SNOMED CT*



Selected *SNOMED CT* attributes have a hierarchical *relationship* to one another known as “attribute hierarchies”. In an attribute *hierarchy*, one general attribute is the parent of one or more specific *subtypes* of that attribute. *Concepts* defined using the more general attribute can inherit *concepts* modeled with the more specific *subtypes* of that attribute.

6.2.2.2.1 Attribute hierarchies used in modeling Procedures



Three groups of attributes are organized as a simple two-level *hierarchy*. The three top level attributes are | PROCEDURE SITE |, | PROCEDURE DEVICE |, and | PROCEDURE MORPHOLOGY |. Each has a sub-attribute to represent the direct object, and another to represent the indirect object. In addition, | PROCEDURE DEVICE | can be specialized by the attributes | USING DEVICE | and | USING ACCESS DEVICE |.

| PROCEDURE DEVICE | attribute *hierarchy*:

- | PROCEDURE DEVICE |
 - | DIRECT DEVICE |
 - | INDIRECT DEVICE |
 - | USING DEVICE |
 - | USING ACCESS DEVICE |

| PROCEDURE MORPHOLOGY | attribute *hierarchy*:

- | PROCEDURE MORPHOLOGY |
 - | DIRECT MORPHOLOGY |
 - | INDIRECT MORPHOLOGY |

| PROCEDURE SITE | attribute *hierarchy*:

- | PROCEDURE SITE |
 - | PROCEDURE SITE - DIRECT |
 - | PROCEDURE SITE - INDIRECT |

6.2.2.2 Attribute hierarchy used in modeling Clinical Findings



| ASSOCIATED WITH | attribute *hierarchy*:

- | ASSOCIATED WITH |
 - | AFTER |
 - | DUE TO |
 - | CAUSATIVE AGENT |

6.2.2.3 Attributes used to define Clinical Finding concepts



Table 106: Approved Clinical Finding attributes summary

Defining Attribute	Subsumed Attribute	Allowable Values
FINDING SITE		Anatomical or acquired body structure 442083009 (<<)
ASSOCIATED MORPHOLOGY		Morphologically abnormal structure 49755003 (<<)

Defining Attribute	Subsumed Attribute	Allowable Values
ASSOCIATED WITH		Clinical Finding 404684003 (<<) Procedure 71388002 (<<) Event 272379006 (<<) Organism 410607006 (<<) Substance 105590001 (<<) Physical object 260787004 (<<) Physical force 78621006 (<<) Pharmaceutical / biologic product 373873005 (<< Q only) SNOMED CT Concept 138875005 (==)
	CAUSATIVE AGENT	Organism 410607006 (<<) Substance 105590001 (<<) Physical object 260787004 (<<) Physical force 78621006 (<<) Pharmaceutical / biologic product 373873005 (<< Q only) SNOMED CT Concept 138875005 (==)
	DUE TO	Clinical Finding 404684003 (<=) Event 272379006 (<=)
	AFTER	Clinical Finding 404684003 (<<) Procedure 71388002 (<<)
SEVERITY		Severities 272141005 (<=)(< Q)
CLINICAL COURSE		Courses 288524001 (<=)(< Q)
EPISODICITY		Episodicities 288526004 (<=)(< Q)
INTERPRETS		Observable entity 363787002 (<<) Laboratory procedure 108252007 (<<) Evaluation procedure 386053000 (<<)
HAS INTERPRETATION		Findings values 260245000 (<<)
PATHOLOGICAL PROCESS		Autoimmune 263680009 (==) Infectious process 441862004 (<<) Hypersensitivity process 472963003 (< <)

Defining Attribute	Subsumed Attribute	Allowable Values
HAS DEFINITIONAL MANIFESTATION		Clinical finding 404684003 (<<)
OCCURRENCE		Periods of life 282032007 (<)
FINDING METHOD		Procedure 71388002 (=<)
FINDING INFORMER		Performer of method 420158005 (<<) Subject of record or other provider of history 419358007 (<<)

👉 Note:

Meaning of Allowable Values (*Range*) notations:

- (<<) this code and *descendants*,
- (<) *descendants* only,
- (<=) *descendants* only (stated) except for supercategory groupers,
- (==) this code only,
- (< Q) *descendants* only when in a qualifying *Relationship*,
- (< Q only) *descendants* only, and only allowed in a qualifying *Relationship*.

👉 Note: See also [Clinical finding](#).

6.2.2.3.1 FINDING SITE



This attribute specifies the body site affected by a condition.

Table 107: Permissible values for FINDING SITE

Attribute Values	Examples
Anatomical or acquired body structure 442083009 (<<)	Kidney disease (disorder) • FINDING SITE Kidney structure (body structure)
	Appendicitis (disorder) • FINDING SITE Appendix structure (body structure)

6.2.2.3.2 ASSOCIATED MORPHOLOGY



This attribute specifies the morphologic changes seen at the tissue or cellular level that are characteristic features of a disease.

Table 108: Permissible values for ASSOCIATED MORPHOLOGY

Attribute Values	Examples
Morphologically abnormal structure 49755003 (<<)	Bone marrow hyperplasia (disorder) • ASSOCIATED MORPHOLOGY Hyperplasia (morphologic abnormality)
	Pancreatitis (disorder) • ASSOCIATED MORPHOLOGY Inflammation (morphologic abnormality)

6.2.2.3.3 ASSOCIATED WITH

This attribute asserts an interaction between two *concepts* beyond simple co-occurrence in the patient. | ASSOCIATED WITH | represents a clinically relevant association between *concepts* without either asserting or excluding a causal or sequential *relationship* between the two.

Table 109: Permissible values for ASSOCIATED WITH

Attribute Values	Examples
Clinical Finding 404684003 (<<) Procedure 71388002 (<<) Event 272379006 (<<) Organism 410607006 (<<) Substance 105590001 (<<) Physical object 260787004 (<<) Physical force 78621006 (<<) Pharmaceutical / biologic product 373873005 (<< Q only) SNOMED CT Concept 138875005 (==)	

| ASSOCIATED WITH | subsumes the following, more specific, attributes in what is called an attribute *hierarchy* (explained in [Attribute Hierarchies in SNOMED CT](#) on page 217):

- | AFTER |
- | DUE TO |
- | CAUSATIVE AGENT |

6.2.2.3.4 AFTER

This attribute is used to model *concepts* in which a clinical finding occurs after another clinical finding or procedure. Neither asserting nor excluding a causal *relationship*, it instead emphasizes a sequence of events.

Table 110: Permissible values for AFTER

Attribute Values	Examples
Clinical Finding 404684003 (<<) Procedure 71388002 (<<)	Post-viral disorder (disorder) • AFTER Viral disease (disorder)

This example can be paraphrased as: “every post-viral disorder occurs after some viral disease”.

6.2.2.3.5 DUE TO



This attribute is used to relate a | Clinical finding | directly to its cause. If a clinical finding merely predisposes to or worsens another disorder, rather than causing it directly, then the more general attribute | ASSOCIATED WITH | is used instead.

Table 111: Permissible values for DUE TO

Attribute Values	Examples
Clinical Finding 404684003 (<=) Event 272379006 (<=)	Cheilitis due to atopic dermatitis (disorder) • IS A Cheilitis (disorder) • DUE TO Atopic dermatitis (disorder)

6.2.2.3.6 CAUSATIVE AGENT



This attribute identifies the direct causative agent of a disease. It does not include vectors, e.g. a mosquito that transmits malaria.

Table 112: Permissible values for CAUSATIVE AGENT

Attribute Values	Examples
Organism 410607006 (<<) Substance 105590001 (<<) Physical object 260787004 (<<) Physical force 78621006 (<<) Pharmaceutical / biologic product 373873005 (<< Q only) SNOMED CT Concept 138875005 (==)	Bacterial endocarditis (disorder) • CAUSATIVE AGENT Superkingdom Bacteria (organism)
	Fentanyl allergy (disorder) • CAUSATIVE AGENT Fentanyl (substance)
	Electrical burn of skin (disorder) • CAUSATIVE AGENT Electricity (physical force)

6.2.2.3.7 SEVERITY



This attribute is used to subclass a | Clinical finding | concept according to its severity; however, caution is encouraged because this use is said to be *relative*. By relative, it is meant that it is incorrect to assume that the same degree of disease intensity or hazard is implied for all | Clinical finding | to which this attribute is applied. There are three *reasons*.

First, “severe” could be interpreted differently depending on what other values are available to choose for severity. Thus severity is relative to the other values in the *value set* presented to users. Consider the different meaning of severity in each of the following three sets of values:

- mild / moderate / severe

- minimal / mild / moderate / severe / very severe
- mild / mild to moderate / moderate / moderate to severe / severe / life threatening / fatal

Second, the severity is defined relative to the expected degree of intensity or hazard of the | Clinical finding | that is being qualified. A common cold has a baseline intensity or hazard much less than that of a more serious disease like lupus erythematosus or pneumonia; thus a severe cold might be considered less intense or hazardous than a mild pneumonia.

Third, some disorders that are life-threatening do not ordinarily have a severity assigned to them. Cancer, for example, is generally not subclassed according to mild, moderate and severe types, but rather is subclassed according to stage or grade.

For these *reasons*, the | SEVERITY | attribute cannot be relied on to retrieve all *Clinical findings* with serious or life-threatening import. Nevertheless, it is still useful for subclassing certain *concepts* and differentiating between different severities of a single disorder. SEVERITY is not used to model any *concepts precoordinated* in the *International Release* but it can still be used in *postcoordination* as a *qualifier*.

Table 113: Permissible values for SEVERITY

Attribute Values	Examples
Severities 272141005 (<=)(< Q)	

6.2.2.3.8 CLINICAL COURSE



This attribute is used to represent both the course and onset of a disease. Many conditions with an acute (sudden) onset also have an acute (short duration) course. Few diseases with a chronic (long - term) course would need to have their onset sub-divided into rapid or gradual *subtypes*, and thus there is no clear need for separating the rapidity of onset from the duration of a disease; based on testing by implementers and *modelers*, a single attribute with values that combine these meanings has clearly been more reproducible and useful than two attributes that attempt to separate the meanings.

Table 114: Permissible values for CLINICAL COURSE

Attribute Values	Examples
Courses 288524001 (<=)(< Q)	<p> Acute amebic dysentery (disorder) </p> <ul style="list-style-type: none"> • CLINICAL COURSE Sudden onset AND/OR short duration (qualifier value) <p> Chronic fibrosing pancreatitis (disorder) </p> <ul style="list-style-type: none"> • CLINICAL COURSE Chronic (qualifier value)

The word acute has more than one meaning, and the meanings are often overlapping or unclear. The word acute may imply rapid onset, short duration, or high severity; in some circumstances it might be used to mean all of these. For morphological *terms* it may also imply the kind of morphology associated with the speed of onset. | Acute inflammation (morphologic abnormality) | does not necessarily have CLINICAL COURSE | Sudden onset AND/OR short duration |, but rather implies polymorphonuclear infiltration; likewise | Chronic inflammation (morphologic abnormality) | implies mononuclear cell infiltration, not necessarily a chronic course, although inflammation with a chronic course is highly correlated with a lymphocytic infiltration.

6.2.2.3.9 EPISODICITY



| EPISODICITY | is used to represent episodes of care provided by a physician or other care provider, typically a general practitioner, *not* episodes of disease experienced by the patient. See [EPISODICITY](#)

no longer modeled in active content on page 449, regarding the origin of the attribute. For example, asthma with | EPISODICITY |=| first episode | represents the first *time* the patient presents to their health care provider with asthma. EPISODICITY is not used to model any *concepts preordinated* in the *International Release* but it can still be used in *postcoordination* as a *qualifier*.

Table 115: Permissible values for EPISODICITY

Attribute Values	Examples
Episodicities 288526004 (<=)(< Q)	

6.2.2.3.10 INTERPRETS

This attribute refers to the entity being evaluated or interpreted, when an evaluation, interpretation or “judgment” is intrinsic to the meaning of a *concept*. This attribute is usually grouped with the | HAS INTERPRETATION | attribute.

Table 116: Permissible values for INTERPRETS

Attribute Values	Examples
Observable entity 363787002 (<<) Laboratory procedure 108252007 (<<) Evaluation procedure 386053000 (<<)	Decreased muscle tone (finding) • INTERPRETS muscle tone (observable entity) • HAS INTERPRETATION Decreased (qualifier value)
	Abnormal glucose level (finding) • INTERPRETS Glucose measurement (procedure) • HAS INTERPRETATION Outside reference range (qualifier value)

 **Note:** For *concepts* in the Measurement finding subhierarchy, the value for | INTERPRETS | should be an Evaluation procedure or a Laboratory procedure rather than an Observable entity.

6.2.2.3.11 HAS INTERPRETATION

This attribute is grouped with the attribute | INTERPRETS |, and designates the judgment aspect being evaluated or interpreted for a *concept* (e.g., presence, absence, degree, normality, abnormality, etc.).

Table 117: Permissible values for HAS INTERPRETATION

Attribute Values	Examples
Findings values 260245000 (<<)	Decreased muscle tone (finding) <ul style="list-style-type: none"> • INTERPRETS Muscle tone (observable entity) • HAS INTERPRETATION Decreased (qualifier value)
	Abnormal glucose level (finding) <ul style="list-style-type: none"> • INTERPRETS Glucose measurement (procedure) • HAS INTERPRETATION Outside reference range (qualifier value)

6.2.2.3.12 PATHOLOGICAL PROCESS

This attribute provides information about the underlying pathological process for a disorder, but only when the results of that process are not structural and cannot be represented by the | ASSOCIATED MORPHOLOGY | attribute.

The values | Infectious process (qualifier value) | and its subtype | Parasitic process (qualifier value) | are included in the range for | PATHOLOGICAL PROCESS |. These were added to accommodate the change in the modeling of *concepts* in the | Infectious disease (disorder) | subhierarchy where the infectious aspect of the disease is represented using | PATHOLOGICAL PROCESS |.

Table 118: Permissible values for PATHOLOGICAL PROCESS

Attribute Values	Examples
Autoimmune 263680009 (==) Infectious process 441862004 (<<) Hypersensitivity process 472963003 (< <)	Autoimmune parathyroiditis (disorder) <ul style="list-style-type: none"> • PATHOLOGICAL PROCESS Autoimmune (qualifier value)
	Disease caused by parasite (disorder) <ul style="list-style-type: none"> • PATHOLOGICAL PROCESS Parasitic process (qualifier value)

Pathological process must not be used for values that could overlap with | ASSOCIATED MORPHOLOGY |. Inflammatory processes result in inflammation (by definition), but these disorders should be defined using their morphology.

6.2.2.3.13 HAS DEFINITIONAL MANIFESTATION

This attribute links disorders to the manifestations (observations) that define them. It can only be applied to disorders.

Table 119: Permissible values for HAS DEFINITIONAL MANIFESTATION

Attribute Values	Examples
Clinical finding 404684003 (<<)	Seizure disorder (disorder) • HAS DEFINITIONAL MANIFESTATION Seizure (finding)
	Hypertensive disorder, systemic arterial (disorder) • HAS DEFINITIONAL MANIFESTATION Finding of increased blood pressure (finding)

6.2.2.3.14 OCCURRENCE

This attribute refers to the specific period of life during which a condition first presents. Multiple values of | OCCURRENCE | for a single *concept* are not desirable, and these will be addressed in a future release. This does not mean the condition cannot persist beyond the period of life in which it first presents.

Table 120: Permissible values for OCCURRENCE

Attribute Values	Examples
Periods of life 282032007 (<)	Childhood phobic anxiety disorder (disorder) • OCCURRENCE Childhood (qualifier value)

6.2.2.3.15 FINDING METHOD

This attribute specifies the means by which a clinical finding was determined. This attribute is frequently used in conjunction with | FINDING INFORMER |. Findings that specify that they were determined by examination of the patient (e.g. | On examination - ankle clonus (finding) |) should have a value for both | FINDING METHOD | and | FINDING INFORMER |.

Table 121: Permissible values for FINDING METHOD

Attribute Values	Examples
Procedure 71388002 (<=)	Finding by palpation (finding) • FINDING METHOD Palpation (procedure)

6.2.2.3.16 FINDING INFORMER

This attribute specifies the person or other entity from which the clinical finding information was obtained. This attribute is frequently used in conjunction with | FINDING METHOD |.

Table 122: Permissible values for FINDING INFORMER

Attribute Values	Examples
Performer of method 420158005 (<<) Subject of record or other provider of history 419358007 (<<)	Complaining of a headache (finding) • FINDING INFORMER Subject of record or other provider of history (person)
	On examination - ankle clonus (finding) • FINDING INFORMER Performer of method (person)

It is accepted that an information model should permit identification of a particular individual who provides information; | FINDING INFORMER | is not about the particular individual. It is about the *category or type* of informer, which is used to differentiate self-reported symptoms from provider-observed signs. Granted, this permits inclusion of epistemology-loaded *terms* (cf. Bodenreider el al., FOIS 2004), but health care is full of such *terms*, and they are (or at least can be) understandable, reproducible and useful.

6.2.2.4 Attributes used to define Procedure concepts

**Table 123: Approved Procedure attributes summary**

Defining Attribute	Subsumed Attribute	Allowable Values
PROCEDURE SITE	Anatomical or acquired body structure 442083009 (<<)	
	Procedure site - Direct	Anatomical or acquired body structure 442083009 (<<)
	Procedure site - Indirect	Anatomical or acquired body structure 442083009 (<<)
PROCEDURE MORPHOLOGY	Morphologically abnormal structure 49755003 (<<)	
	Direct morphology	Morphologically abnormal structure 49755003 (<<)
	Indirect morphology	Morphologically abnormal structure 49755003 (<<)
METHOD	Action 129264002 (<<)	

Defining Attribute	Subsumed Attribute	Allowable Values
PROCEDURE DEVICE	Device 49062001 (<<)	
	DIRECT DEVICE	Device 49062001 (<<)
	INDIRECT DEVICE	Device 49062001 (<<)
	USING DEVICE	Device 49062001 (<<)
	USING ACCESS DEVICE	Device 49062001 (<<)
ACCESS		Surgical access values 309795001 (<=)(< Q)
DIRECT SUBSTANCE	Substance 105590001 (<<)	
		Pharmaceutical / biologic product 373873005 (<<)
PRIORITY		Priorities 272125009 (<=)(< Q)
HAS FOCUS	Clinical finding 404684003 (<<)	
		Procedure 71388002 (<<)
HAS INTENT		Intents (nature of procedure values) 363675004 (<=)
RECIPIENT CATEGORY	Person 125676002 (<<)	
	Family 35359004 (<<)	
	Community 133928008 (<<)	
	Donor for medical or surgical procedure 105455006 (<<)	
	Group 389109008 (<<)	
REVISION STATUS	Primary operation 261424001 (<<)	
	Revision - value 255231005 (<<)	
	Part of multistage procedure 257958009 (<<)	
ROUTE OF ADMINISTRATION		Route of administration value 284009009 (<<)
SURGICAL APPROACH		Procedural approach 103379005 (<=)(< Q)
USING ENERGY		Physical force 78621006 (<<)
USING SUBSTANCE		Substance 105590001 (<<)

 **Note:**

Meaning of Allowable Values (*Range*) notations:

- (<<) this code and *descendants*,
- (<) *descendants* only,
- (<=) *descendants* only (stated) except for supercategory groupers,
- (==) this code only,
- (< Q) *descendants* only when in a qualifying *Relationship*,
- (< Q only) *descendants* only, and only allowed in a qualifying *Relationship*.

 **Note:**

Attributes should be grouped with the | METHOD | attribute to which they apply; in the absence of a | METHOD | attribute, attributes that are related to each other should be grouped. The one exception is | RECIPIENT CATEGORY |, because a single procedure code should not be *precoordinated* in situations where more than one recipient category is involved. Such complex statements should utilize two or more procedure codes that are placed into an appropriately structured information model.

 **Note:** See also [Procedure](#).

6.2.2.4.1 PROCEDURE SITE



The | PROCEDURE SITE | attribute describes the body site acted on or affected by a procedure.

This attribute subsumes, in an attribute *hierarchy* (see [Attribute Hierarchies in SNOMED CT](#) on page 217), the more specific attributes (| Procedure site - Direct | and | Procedure site - Indirect |) that should be used if possible. The anatomical site may be directly acted on (| Procedure site - Direct |) or indirectly acted upon (| Procedure site - Indirect |).

When modeling procedures where the | METHOD | is | Removal - action | or one of its *subtypes* (e.g. | Excision |, | Surgical biopsy |, etc.), removals **of** the structure itself should use | Procedure site - Direct |. Removals of tissue lesions (cysts, tumors, etc.) are considered to be removals of the site, and should also use | Procedure site - Direct |. Removals of devices, calculi, thrombi, foreign bodies and other non-tissue entities **from** the structure should use | Procedure site - Indirect |.

Table 124: Permissible values for PROCEDURE SITE

Attribute Values	Examples
Anatomical or acquired body structure 442083009 (<<)	Procedure on colon (procedure) • PROCEDURE SITE colon structure (body structure)

Procedures need not necessarily be categorized by site. | Human body structure | should *not* be assigned as a default value of this attribute because many procedures can be performed on non-human subjects, and because this attribute does *not* necessarily need to be present in a procedure *concept* definition in order for *classifier* algorithms to work properly.

The general | PROCEDURE SITE | attribute is used to model the site for high-level grouper type procedure *concepts*. It is most likely to be used for *concepts* that do not require a | METHOD | (action) attribute. Relatively few *concepts* will be modeled using | PROCEDURE SITE |, rather than the more specific direct and indirect site attributes (see below).

6.2.2.4.1.1 PROCEDURE SITE DIRECT



This attribute is used when the action of the procedure is directly aimed at an anatomical or acquired body structure or site rather than at something else (such as a device) located there.

Table 125: Permissible values for PROCEDURE SITE DIRECT

Attribute Values	Examples
Anatomical or acquired body structure 442083009 (<<)	Amputation of the foot (procedure) <ul style="list-style-type: none"> • METHOD Amputation - action (qualifier value) • Procedure site - Direct Foot structure (body structure)
	Biopsy of femur (procedure) <ul style="list-style-type: none"> • METHOD Biopsy - action (qualifier value) • Procedure site - Direct Bone structure of femur (body structure)

6.2.2.4.1.1.1 Multiple values for PROCEDURE SITE DIRECT



When the | METHOD | (action) acts directly on a morphological abnormality (more simply, a lesion) arising from, or existing in, the cells of the tissue in which it occurs [e.g. a tumor (including metastatic tumors), granuloma, polyp, or cyst] the attribute | DIRECT MORPHOLOGY | is used to model the morphological abnormality. Most *concept* definitions where | DIRECT MORPHOLOGY | is used, which also require a site in the definition, will use | Procedure site - Direct |. Thus, there can be more than one direct object of the | METHOD | for a *concept*. For example, the | DIRECT MORPHOLOGY | and the | Procedure site - Direct | can both be direct objects of the | METHOD |. An example of an exception to this rule would be removal of a calculus from the ureter. In this case, the calculus is the direct object, but there is no procedure site that is that direct object, since the ureter is an indirect object.

The most common *concepts* that have more than one direct object of the | METHOD | are *Subtypes* of | Removal (procedure) | where the object of the removal (e.g. a neoplasm) can be considered to be a part of the tissue at the anatomical site in which it occurs. When a part of an anatomical structure (however abnormal) has been removed, both the morphological abnormality and the anatomical structure in which it is located are to be modeled as direct objects for the | METHOD || Removal - action (qualifier value) |. Grafts that become attached via in-growth of capillaries, fibroblasts, and/or other cells or tissues would also be regarded as biologically connected, and therefore modeling their removal would include the anatomical structure as a direct object of the action. The anatomical structure is not to be modeled as a direct object of a removal only when the procedure does not necessarily involve removal also of part of the anatomy; examples include removals of things such as a foreign body, a catheter, a renal calculus, or a mechanical implant like a pacemaker.

6.2.2.4.1.2 PROCEDURE SITE INDIRECT



This attribute describes the anatomical site, which is acted upon, but is not the direct object of the procedure. (The site is indirectly acted on by the procedure.) Usually in these procedures there is another value that is the direct object of the action. Exceptions (*concepts* that do not specify a direct object, but only an indirect object) are usually general groupers such as | Arm implantation (procedure) | (meaning implantation of something into the arm), since the thing implanted could be either a device or a substance (material).

Table 126: Permissible values for PROCEDURE SITE INDIRECT

Attribute Values	Examples
Anatomical or acquired body structure 442083009 (<<)	Removal of catheter from brachial vein (procedure) • METHOD Removal - action (qualifier value) • DIRECT DEVICE Catheter, device (physical object) • Procedure site - Indirect Structure of brachial vein (body structure)
	Removal of calculus of urinary bladder (procedure) • METHOD Removal - action (qualifier value) • DIRECT MORPHOLOGY Calculus (morphologic abnormality) • Procedure site - Indirect Urinary bladder structure (body structure)

6.2.2.4.2 PROCEDURE MORPHOLOGY

| PROCEDURE MORPHOLOGY | is the attribute used to specify the morphology or abnormal structure involved in a procedure. This attribute subsumes the more specific attributes | DIRECT MORPHOLOGY | and | INDIRECT MORPHOLOGY | that should be used if possible (see below). | DIRECT MORPHOLOGY | is used when the procedure method acts directly on the morphologic abnormality. | INDIRECT MORPHOLOGY | is used when the procedure method acts directly on something else (e.g. a device, substance or anatomical structure) that is associated with the morphologic abnormality. The more general attribute | PROCEDURE MORPHOLOGY | is used when defining general *concepts* that subsume both kinds of sub-concepts.

Table 127: Permissible values for PROCEDURE MORPHOLOGY

Attribute Values	Examples
Morphologically abnormal structure 49755003 (<<)	

Hematoma , calculus, foreign body, blood clot, embolus, and some other entities are not strictly body structures, but are in the body structure *hierarchy* under morphologically abnormal structure, and are valid values for the | PROCEDURE MORPHOLOGY | attributes.

6.2.2.4.2.1 DIRECT MORPHOLOGY

This attribute describes the morphologically abnormal structure that is the direct object of the METHOD action.

Table 128: Permissible values for DIRECT MORPHOLOGY

Attribute Values	Examples
Morphologically abnormal structure 49755003 (<<)	Excision of benign neoplasm (procedure) • METHOD Excision - action (qualifier value) • DIRECT MORPHOLOGY Neoplasm, benign (morphologic abnormality)

6.2.2.4.2.2 INDIRECT MORPHOLOGY



This attribute represents a morphology that is acted upon, but is not the direct target of the action being performed (i.e. the procedure's method acts directly on something else, such as a device, substance, or anatomical structure).

Table 129: Permissible values for INDIRECT MORPHOLOGY

Attribute Values	Examples
Morphologically abnormal structure 49755003 (<<)	<ul style="list-style-type: none"> Removal of mesh from wound (procedure) <ul style="list-style-type: none"> • METHOD Removal - action (qualifier value) • DIRECT DEVICE Mesh (physical object) • INDIRECT MORPHOLOGY Wound (morphologic abnormality)

6.2.2.4.3 METHOD



This attribute represents the action being performed to accomplish the procedure. It does not include the surgical approach (e.g. translumbar), equipment (e.g. sutures), or physical forces (e.g. laser energy).

Table 130: Permissible values for METHOD

Attribute Values	Examples
Action 129264002 (<<)	<ul style="list-style-type: none"> Incision of ureter (procedure) <ul style="list-style-type: none"> • METHOD Incision - action (qualifier value) • Procedure site - Direct Ureretic structure (body structure)

The | METHOD | can be considered the anchor of each *relationship group* that defines a procedure; if there are two methods, there should be two different *relationship groups*. It is correct to regard each *relationship group* as a kind of sub-procedure that defines the overall procedure. Each method can be regarded as the verb of a sentence, and the verbs direct and indirect objects are specified by the site, morphology, device, substance or energy attributes (below) that are grouped with it.

6.2.2.4.4 PROCEDURE DEVICE



| PROCEDURE DEVICE | is a general attribute used to model devices associated with a procedure. It subsumes the more specific attributes | DIRECT DEVICE |, | INDIRECT DEVICE |, | USING DEVICE |, and | USING ACCESS DEVICE |, which should be used instead of | PROCEDURE DEVICE | if possible. The general attribute | PROCEDURE DEVICE | is mainly useful for defining high-level, general concepts that aggregate procedures according to the device involved.

Table 131: Permissible values for PROCEDURE DEVICE

Attribute Values	Examples
Device 49062001 (<<)	<ul style="list-style-type: none"> Catheter procedure (procedure) <ul style="list-style-type: none"> • PROCEDURE DEVICE Catheter, device (physical object)

When the device is the direct object of the action (| METHOD |), the attribute | DIRECT DEVICE | is used. If the action is done indirectly to the device, that is, the action is done to something that is located in or on a

device, but is not done directly to the device itself, then the attribute | INDIRECT DEVICE | is used. If the device is used to carry out the action, then the attribute | USING DEVICE | is used. If the device is used to access the site of the action, then the attribute | USING ACCESS DEVICE | is used.

 **Note:** The permissible values for attributes in the | PROCEDURE DEVICE | role *hierarchy* include | Device (physical object) | and its *descendants*. However, there are a limited number of products in SNOMED CT which are devices that also deliver drugs. These *concepts* descend from | Drug-device combination product (product) | which is a *descendant* of both | Device (physical object) | and | Pharmaceutical / biologic product (product) |. Therefore, although they carry the *hierarchy* tag of (product), they are valid values for attributes in the | PROCEDURE DEVICE | role *hierarchy*.

Example:

- | Removal of drug coated stent (procedure) |
 - | METHOD | | Removal - action (qualifier value) |
 - | DIRECT DEVICE | | Drug coated stent (product) |

6.2.2.4.4.1 DIRECT DEVICE



This attribute represents the device on which the method directly acts.

Table 132: Permissible values for DIRECT DEVICE

Attribute Values	Examples
Device 49062001 (<<)	Removal of arterial stent (procedure) <ul style="list-style-type: none"> • METHOD Removal - action (qualifier value) • DIRECT DEVICE Arterial stent (physical object)

6.2.2.4.4.2 INDIRECT DEVICE



This attribute models action done on something that is located in or on a device, but is not done directly on the device itself.

Table 133: Permissible values for INDIRECT DEVICE

Attribute Values	Examples
Device 49062001 (<<)	Excision of vegetations from implanted mitral valve (procedure) <ul style="list-style-type: none"> • METHOD Excision - action (qualifier value) • DIRECT MORPHOLOGY Vegetation (morphologic abnormality) • INDIRECT DEVICE Mitral valve prosthesis, device (physical object) • Procedure site - Indirect Mitral valve structure (body structure)

 **Note:**

In the above example, the vegetation is being excised. The mitral valve prosthesis is where the excised vegetation is located but the mitral valve prosthesis itself is not excised. Thus, mitral valve prosthesis is the | INDIRECT DEVICE |.

 **Note:**

The attribute | INDIRECT DEVICE | is infrequently needed. When using this attribute, a second look is advisable to be sure it is needed.

6.2.2.4.4.3 USING DEVICE



This attribute refers to the instrument or equipment utilized to execute an action. | USING DEVICE | is used when the device is actually used to carry out the action that is the focus of the procedure. If the device is simply the means to access the site of the procedure, then | USING ACCESS DEVICE | is used instead of | USING DEVICE |.

Table 134: Permissible values for USING DEVICE

Attribute Values	Examples
Device 49062001 (<<)	Core needle biopsy of larynx (procedure) <ul style="list-style-type: none"> • METHOD Biopsy - action (qualifier value) • USING DEVICE Core biopsy needle, device (physical object) • Procedure site - Direct Laryngeal structure (body structure)

6.2.2.4.4.4 USING ACCESS DEVICE



This attribute specifies the instrument or equipment used to access the site of a procedure.

Table 135: Permissible values for USING ACCESS DEVICE

Attribute Values	Examples
Device 49062001 (<<)	Arthroscopic synovial biopsy (procedure) <ul style="list-style-type: none"> • METHOD Biopsy - action (qualifier value) • USING ACCESS DEVICE Arthroscope, device (physical object) • Procedure site - Direct Structure of synovial tissue of joint (body structure)

6.2.2.4.5 ACCESS



This attribute describes the route used to access the site of a procedure. It is used to distinguish open, closed, and percutaneous procedures.

Table 136: Permissible values for ACCESS

Attribute Values	Examples
Surgical access values 309795001 (<=)(< Q)	Open removal of bile duct stent (procedure) <ul style="list-style-type: none"> • ACCESS Open approach - access (qualifier value)

6.2.2.4.6 DIRECT SUBSTANCE



This attribute describes the | Substance | or | Pharmaceutical / biologic product | on which the procedure's method directly acts.

Table 137: Permissible values for DIRECT SUBSTANCE

Attribute Values	Examples
Substance 105590001 (<<) Pharmaceutical / biologic product 373873005 (<<)	Injection of prostaglandin (procedure) • METHOD Injection - action (qualifier value) • DIRECT SUBSTANCE Prostaglandin (substance)

👉 **Note:** As an editorial policy, in the distribution form of the *International Release*, | Pharmaceutical / biologic product (product) | and its *descendants* are not used as values for | DIRECT SUBSTANCE |.

6.2.2.4.7 PRIORITY

This attribute refers to the priority assigned to a procedure.

**Table 138: Permissible values for PRIORITY**

Attribute Values	Examples
Priorities 272125009 (<=)(< Q)	Emergency cesarean section (procedure) • PRIORITY Emergency (qualifier value)

6.2.2.4.8 HAS FOCUS

This attribute specifies the | Clinical finding | or | Procedure | which is the focus of a procedure.

**Table 139: Permissible values for HAS FOCUS**

Attribute Values	Examples
Clinical finding 404684003 (<<) Procedure 71388002 (<<)	Cardiac rehabilitation assessment (procedure) • HAS FOCUS Cardiac rehabilitation (regime/therapy)

6.2.2.4.9 HAS INTENT

This attribute specifies the intent of a procedure.

**Table 140: Permissible values for HAS INTENT**

Attribute Values	Examples
Intents (nature of procedure values) 363675004 (<=)	Diagnostic bronchoscopy (procedure) • HAS INTENT Diagnostic intent (qualifier value)

6.2.2.4.10 RECIPIENT CATEGORY



This attribute specifies the type of individual or group upon which the action of the procedure is performed. For example, it can be used in blood banking procedures to differentiate whether the procedure was performed on the donor or the recipient of a blood product. In other words, | RECIPIENT CATEGORY | is | Donor for medical or surgical procedure (person) | if the subject of the record is the donor.

It is not used for a procedure where the subject of the procedure is someone other than the subject of record.

Table 141: Permissible values for RECIPIENT CATEGORY

Attribute Values	Examples
Person 125676002 (<<) Family 35359004 (<<) Community 133928008 (<<) Donor for medical or surgical procedure 105455006 (<<) Group 389109008 (<<)	Social service interview of family (procedure) • RECIPIENT CATEGORY Family (social concept)

6.2.2.4.11 REVISION STATUS

This attribute specifies whether a procedure is primary or a revision.

Table 142: Permissible values for REVISION STATUS

Attribute Values	Examples
Primary operation 261424001 (<<) Revision - value 255231005 (<<) Part of multistage procedure 257958009 (<<)	Primary repair of inguinal hernia (procedure) • REVISION STATUS Primary operation (qualifier value) Revision of knee arthroplasty (procedure) • REVISION STATUS Revision - value (qualifier value)

6.2.2.4.12 ROUTE OF ADMINISTRATION

This attribute allows representation of the route by which a procedure introduces a given substance into the body.

The domain for this attribute is the *sub-hierarchy* below | Administration of substance via specific route (procedure) | 433590000.

Table 143: Permissible values for ROUTE OF ADMINISTRATION

Attribute Values	Examples
Route of administration value 284009009 (<<)	Inhaled drug administration (procedure) • ROUTE OF ADMINISTRATION By inhalation (route) (qualifier value)

6.2.2.4.13 SURGICAL APPROACH

This attribute specifies the directional, relational, or spatial access to the site of a surgical procedure. The domain for | SURGICAL APPROACH | is *descendants* of | Surgical procedure (procedure) | 387713003.

Table 144: Permissible values for SURGICAL APPROACH

Attribute Values	Examples
Procedural approach 103379005 (<=)(< Q)	Intranasal ethmoidectomy (procedure) <ul style="list-style-type: none"> • SURGICAL APPROACH Intranasal approach (qualifier value)
	Abdominal hysterectomy (procedure) <ul style="list-style-type: none"> • SURGICAL APPROACH Abdominal approach (qualifier value)

6.2.2.4.14 USING SUBSTANCE

This attribute describes the | Substance | used to execute the action of a procedure, but it is not the substance on which the procedure's method directly acts (the | DIRECT SUBSTANCE |).

Table 145: Permissible values for USING SUBSTANCE

Attribute Values	Examples
Substance 105590001 (<<)	Contrast radiography of esophagus (procedure) <ul style="list-style-type: none"> • METHOD Radiographic imaging - action (qualifier value) • Procedure site - Direct Esophageal structure (body structure) • USING SUBSTANCE Contrast media (substance)

6.2.2.4.15 USING ENERGY

This attribute describes the energy used to execute an action. | USING ENERGY | has been introduced because the new attribute | USING DEVICE | is now used only to represent the instrument or equipment used to execute the action. Unlike the attribute USING, which it replaces, | USING DEVICE | does not take values from the | physical force | hierarchy.

Table 146: Permissible values for USING ENERGY

Attribute Values	Examples
Physical force 78621006 (<<)	Gamma ray therapy (procedure) <ul style="list-style-type: none"> • USING ENERGY Gamma radiation (physical force)

6.2.2.4.16 Direct and indirect objects

Procedures that have a | METHOD | attribute can be described using an action verb that corresponds to the method. The direct object(s) of the action verb should be represented using (at least) one of the four direct object attributes, depending on whether the direct object on which the method acts is a device (| DIRECT DEVICE |), anatomical structure (| Procedure site - Direct |), morphologic abnormality (| DIRECT MORPHOLOGY |) or substance (| DIRECT SUBSTANCE |).

When the type (body structure, device, or substance) of direct object is indeterminate, the direct-object attributes should not be used.

6.2.2.5 Attributes used to define Evaluation Procedure concepts



Table 147: Approved Evaluation Procedure attributes summary

Defining Attribute	Allowable Values
HAS SPECIMEN	Specimen 123038009 (<=)(< Q)
COMPONENT	Substance 105590001 (<=)(< Q) Observable entity 363787002 (<=)(< Q) Cell structure 4421005 (<=)(< Q) Organism 410607006 (<=)(< Q)
TIME ASPECT	Time frame 7389001 (<=)(< Q)
PROPERTY	Property of measurement 118598001 (<=)(< Q)
SCALE TYPE	Quantitative 30766002 (<<) Qualitative 26716007 (<<) Ordinal value 117363000 (<<) Ordinal or quantitative value 117365007 (<<) Nominal value 117362005 (<<) Narrative value 117364006 (<<) Text value 117444000 (<<)
MEASUREMENT METHOD	Laboratory procedure categorized by method 127789004(<=)

👉 **Note:**

Meaning of Allowable Values (*Range*) notations:

- (<<) this code and *descendants*,
- (<) *descendants* only,
- (<=) *descendants* only (stated) except for supercategory groupers,
- (==) this code only,
- (< Q) *descendants* only when in a qualifying *Relationship*,
- (< Q only) *descendants* only, and only allowed in a qualifying *Relationship*.

👉 **Note:** See also *Observable entity*.

6.2.2.5.1 HAS SPECIMEN



This attribute specifies the type of specimen on which a measurement or observation is performed.

Table 148: Permissible values for HAS SPECIMEN

Attribute Values	
Specimen 123038009 (<=)(< Q)	

6.2.2.5.2 COMPONENT

This attribute refers to what is being observed or measured by a procedure.

Table 149: Permissible values for COMPONENT

Attribute Values	Example
Substance 105590001 (<=)(< Q)	Protein measurement (procedure)
Observable entity 363787002 (<=)(< Q)	• COMPONENT Protein (substance)
Cell structure 4421005 (<=)(< Q)	
Organism 410607006 (<=)(< Q)	

6.2.2.5.3 TIME ASPECT

This attribute specifies temporal *relationships* for a measurement procedure.

Table 150: Permissible values for TIME ASPECT

Attribute Values	
Time frame 7389001 (<=)(< Q)	

6.2.2.5.4 PROPERTY

This attribute specifies the kind of property being measured (e.g. concentration).

Table 151: Permissible values for PROPERTY

Attribute Values	
Property of measurement 118598001 (<=)(< Q)	

6.2.2.5.5 SCALE TYPE

This attribute refers to the scale of the result of an observation of a diagnostic test (i.e. quantitative, qualitative, semi-quantitative).

Table 152: Permissible values for SCALE TYPE

Attribute Values	
Quantitative 30766002 (<<)	
Qualitative 26716007 (<<)	
Ordinal value 117363000 (<<)	
Ordinal or quantitative value 117365007 (<<)	
Nominal value 117362005 (<<)	
Narrative value 117364006 (<<)	
Text value 117444000 (<<)	

6.2.2.5.6 MEASUREMENT METHOD

This attribute specifies the method by which a procedure is performed.

Table 153: Permissible values for MEASUREMENT METHOD

Attribute Values	
Laboratory procedure categorized by method 127789004(<=)	

For measurement procedures, the attribute | METHOD | is given the value | Measurement - action (qualifier value) |. The attribute | MEASUREMENT METHOD | can be used to provide additional specificity.

6.2.2.6 Attributes used to define Specimen concepts**Table 154: Approved Specimen attributes summary**

Defining Attribute	Allowable Values
SPECIMEN PROCEDURE	Procedure 71388002 (<)
SPECIMEN SOURCE TOPOGRAPHY	Anatomical or acquired body structure 442083009 (<<)
SPECIMEN SOURCE MORPHOLOGY	Morphologically abnormal structure 49755003 (<<)
SPECIMEN SUBSTANCE	Substance 105590001 (<<)
SPECIMEN SOURCE IDENTITY	Person 125676002 (<<) Family 35359004 (<<) Community 133928008 (<<) Device 49062001 (<<) Environment 276339004 (<<)

 **Note:**

Meaning of Allowable Values (*Range*) notations:

- (<<) this code and *descendants*,
- (<) *descendants* only,
- (<=) *descendants* only (stated) except for supercategory groupers,
- (==) this code only,
- (< Q) *descendants* only when in a qualifying *Relationship*,
- (< Q only) *descendants* only, and only allowed in a qualifying *Relationship*.

 **Note:** See also [Specimen](#).

6.2.2.6.1 SPECIMEN PROCEDURE



This attribute identifies the procedure by which a specimen is obtained.

Table 155: Permissible values for SPECIMEN PROCEDURE

Attribute Values	Examples
Procedure 71388002 (<)	<ul style="list-style-type: none"> Urine specimen obtained by clean catch procedure (specimen) <ul style="list-style-type: none"> • SPECIMEN PROCEDURE Urine specimen collection, clean catch (procedure)
	<ul style="list-style-type: none"> Specimen from stomach obtained by total gastrectomy (specimen) <ul style="list-style-type: none"> • SPECIMEN PROCEDURE Total gastrectomy (procedure)

6.2.2.6.2 SPECIMEN SOURCE TOPOGRAPHY



This attribute specifies the body site from which a specimen is obtained.

Table 156: Permissible values for SPECIMEN SOURCE TOPOGRAPHY

Attribute Values	Examples
Anatomical or acquired body structure 442083009 (<<)	<ul style="list-style-type: none"> Cervix cytologic material (specimen) <ul style="list-style-type: none"> • SPECIMEN SOURCE TOPOGRAPHY Cervix uteri structure (body structure)
	<ul style="list-style-type: none"> Omentum biopsy sample (specimen) <ul style="list-style-type: none"> • SPECIMEN SOURCE TOPOGRAPHY Omentum structure (body structure)

6.2.2.6.3 SPECIMEN SOURCE MORPHOLOGY



This *attribute* names the morphologic abnormality from which a specimen is obtained.

Table 157: Permissible values for SPECIMEN SOURCE MORPHOLOGY

Attribute Values	Examples
Morphologically abnormal structure 49755003 (<<)	Specimen from cyst (specimen) <ul style="list-style-type: none"> • SPECIMEN SOURCE MORPHOLOGY Cyst (morphologic abnormality)
	Specimen from wound abscess (specimen) <ul style="list-style-type: none"> • SPECIMEN SOURCE MORPHOLOGY Abscess of wound (morphologic abnormality)

6.2.2.6.4 SPECIMEN SUBSTANCE

This *attribute* names the type of substance of which a specimen is comprised.

Table 158: Permissible values for SPECIMEN SUBSTANCE

Attribute Values	Examples
Substance 105590001 (<<)	Mid-stream urine sample (specimen) <ul style="list-style-type: none"> • SPECIMEN SUBSTANCE Urine (substance)
	Pancreatic fluid specimen (specimen) <ul style="list-style-type: none"> • SPECIMEN SUBSTANCE Pancreatic fluid (substance)

6.2.2.6.5 SPECIMEN SOURCE IDENTITY

This *attribute* names the type of individual, group, or physical location from which a specimen is collected.

Table 159: Permissible values for SPECIMEN SOURCE IDENTITY

Attribute Values	Examples
Person 125676002 (<<) Family 35359004 (<<) Community 133928008 (<<) Device 49062001 (<<) Environment 276339004 (<<)	Blood specimen from blood donor (specimen) <ul style="list-style-type: none"> • SPECIMEN SOURCE IDENTITY Blood donor (person)
	Catheter tip specimen (specimen) <ul style="list-style-type: none"> • SPECIMEN SOURCE IDENTITY Catheter tip, device (physical object)

6.2.2.7 Attributes used to define Body structure concepts

Just one attribute is used in Anatomy, namely, | Laterality | .

 **Note:** See also [Body structure](#).

6.2.2.7.1 LATERALITY



This attribute provides information on whether a body structure is left, right, bilateral or unilateral. It is applied only to bilaterally symmetrical body structures which exist on opposite sides of the body.

Table 160: Permissible values for LATERALITY

Attribute Values	Examples
Side 182353008 (<=)	Left kidney structure (body structure) • LATERALITY Left (qualifier value)

 **Note:**

Permissible values for this attribute include the *descendants* of the *concept* listed, except for super category grouper *concepts*.

6.2.2.8 Attributes used to define Pharmaceutical/Biologic Product concepts



Table 161: Approved Pharmaceutical/Biologic Product attributes summary

Defining Attribute	Allowable Values
HAS ACTIVE INGREDIENT	Substance 105590001 (<<)
HAS DOSE FORM	Type of drug preparation 105904009 (<<)

 **Note:**

Permissible values for these attributes include the *concepts* listed and their *descendants*.

 **Note:** See also [Pharmaceutical/biologic product](#).

6.2.2.8.1 HAS ACTIVE INGREDIENT



This attribute indicates the *active ingredient* of a drug product, linking the | Pharmaceutical / biologic product | *hierarchy* to the | Substance | *hierarchy*.

Table 162: Permissible values for HAS ACTIVE INGREDIENT

Attribute Values	Examples
Substance 105590001 (<<)	Naproxen 500mg tablet (product) • HAS ACTIVE INGREDIENT Naproxen (substance)

6.2.2.8.2 HAS DOSE FORM



This attribute specifies the dose form of a product.

Table 163: Permissible values for HAS DOSE FORM

Attribute Values	Examples
Type of drug preparation 105904009 (<<)	Digoxin 0.1mg capsule (product) • HAS DOSE FORM Oral capsule (qualifier value)

6.2.2.9 Attributes used to define Situation with Explicit Context concepts**Table 164: Approved Situation attributes summary**

Defining Attribute	Allowable Values
ASSOCIATED FINDING	Clinical finding 404684003 (<=)(< Q) Event 272379006 (<=)(< Q) Observable entity 363787002 (< Q only) Link assertion 416698001 (< Q only) Procedure 71388002 (< Q only)
FINDING CONTEXT	Finding context value 410514004 (<=)(< Q)
ASSOCIATED PROCEDURE	Procedure 71388002 (<=)(< Q) Observable entity 363787002 (< Q only)
PROCEDURE CONTEXT	Context values for actions 288532009 (<=)(< Q)
TEMPORAL CONTEXT	Temporal context value 410510008 (<=)(< Q)
SUBJECT RELATIONSHIP CONTEXT	Person 125676002 (<=)(< Q)

Note:

Meaning of Allowable Values (*Range*) notations:

- (<<) this code and *descendants*,
- (<) *descendants* only,
- (<=) *descendants* only (stated) except for supercategory groupers,
- (==) this code only,
- (< Q) *descendants* only when in a qualifying *Relationship*,
- (< Q only) *descendants* only, and only allowed in a qualifying *Relationship*.

Note: See also [Situation with explicit context](#).**6.2.2.9.1 Context**

The meaning conveyed by a *SNOMED CT* code in a medical record is affected by the context in which it is recorded. For instance, the code for "breast cancer" might be used to indicate a family history

of breast cancer, a past history of breast cancer, or a *current diagnosis* of breast cancer. Each of these three meanings differs in regard to the context in which breast cancer is being described. Family history of breast cancer refers to breast cancer occurring in a family member of a patient. Past history of breast cancer indicates that the breast cancer occurred in the patient, at some *time* in the past, and it is not necessarily present now. *Current diagnosis* of breast cancer indicates that the breast cancer is present now, and in this patient. These differences are important for data retrieval, because it would be incorrect when searching for patients with breast cancer to retrieve those who merely have a family history of breast cancer.

6.2.2.9.2 Default Context



When a *SNOMED CT* code appears in a record without any explicitly stated context, that code is considered to have a default context. The default is "soft" in that it can be over-ridden by information carried in the structure of the record or its information model.

The default context for a clinical finding code implies that the finding has actually occurred (vs. being absent), that it applies to the subject of the record (the patient), and that it is occurring currently or occurred at a past *time* that is given by a date - *time* record linked to the code.

The default context for a procedure code implies that the procedure was completed, that it was performed on the subject of the record (the patient), and that it was done at the present *time* or in the past at a *time* that is given by a date - *time* record linked to the code.

6.2.2.9.3 Axis Modifiers



The six attributes used to define situation codes permit explicit (rather than default) representation of various contexts. These attributes can change the meaning of a clinical finding or procedure code in a way that changes the *hierarchy* (or "axis") of the code from | Clinical finding | or | Procedure | to | Situation with explicit context |. The resulting modified meaning is not a *subtype* of the original meaning of the code, and therefore the axis-modifying attributes are not used to qualify the code, but instead are used to qualify a "situation" code.

For instance, if | Fine needle biopsy (procedure) | is given the non-context modifying attribute | Procedure site - Direct | and a value of | Urinary bladder structure (body structure) |, the resulting *concept* | Fine needle biopsy of urinary bladder (procedure) | is still a *subtype* of the original *concept* | Fine needle biopsy (procedure) |.

However, the *concept* | Urine protein test not done (situation) | uses the context-modifying attribute | PROCEDURE CONTEXT | and a value of | Not done (qualifier value) |, and the resulting *concept* is not a *subtype* of | Urine protein test (procedure) |. Its axis (*hierarchy*) has been modified.

6.2.2.9.4 Overview of context attributes



Of the six attributes applied to *concepts* in the | Situation with explicit context | *hierarchy*, two are used only in representing the context in which a | Clinical finding | is recorded, (| ASSOCIATED FINDING | and | FINDING CONTEXT |); two are used only in representing the context in which a | Procedure | is recorded (| ASSOCIATED PROCEDURE | and | PROCEDURE CONTEXT |); and two attributes are used in representing the context of both | Procedure | and | Clinical finding | (| SUBJECT RELATIONSHIP CONTEXT | and | TEMPORAL CONTEXT |).

6.2.2.9.5 ASSOCIATED FINDING



This attribute links *concepts* in the | Situation with explicit context | *hierarchy* to their related | Clinical finding |. It specifies the | Clinical finding | *concept* whose context is being modified.

Table 165: Permissible values for ASSOCIATED FINDING

Attribute Values	Examples
Clinical finding 404684003 (<=)(< Q) Event 272379006 (<=)(< Q) Observable entity 363787002 (< Q only) Link assertion 416698001 (< Q only) Procedure 71388002 (< Q only)	Family history of stroke (situation) • ASSOCIATED FINDING Cerebrovascular accident (disorder)

 **Note:**

When | ASSOCIATED FINDING | is used in *postcoordinated expressions*, its range is broader than when used in distributed content.

| ASSOCIATED FINDING | must not reference *concepts* that already have *precoordinated context* themselves.

For example, the following definition uses | FH: Thyroid disorder | incorrectly:

| History of thyroid disease in father |:

- | SUBJECT RELATIONSHIP CONTEXT |=| father |
- | ASSOCIATED FINDING |=| FH: Thyroid disorder |.

The following is the correct definition:

| History of thyroid disease in father |:

- | SUBJECT RELATIONSHIP CONTEXT |=| father |
- | ASSOCIATED FINDING |=| thyroid disease |.

6.2.2.9.6 FINDING CONTEXT



The FINDING CONTEXT attribute is used to represent a situation in which a *Clinical finding* is known or unknown, and if known, whether it is present, absent, or uncertain (possible); and also to express the meaning that the finding is not actual but instead an anticipated or possible future finding.

Table 166: Permissible values for FINDING CONTEXT

Attribute Values	Examples
Finding context value 410514004 (<=)(< Q)	No cough (situation) • ASSOCIATED FINDING Cough (finding) • FINDING CONTEXT Known absent (qualifier value)

6.2.2.9.7 ASSOCIATED PROCEDURE



This attribute links *concepts* in the | Situation with explicit context | *hierarchy* to *concepts* in the | Procedure | *hierarchy* for which there is additional specified context.

Table 167: Permissible values for ASSOCIATED PROCEDURE

Attribute Values	Examples
Procedure 71388002 (<=)(< Q) Observable entity 363787002 (< Q only)	Operative procedure planned (situation) • ASSOCIATED PROCEDURE Surgical procedure (procedure)

6.2.2.9.8 PROCEDURE CONTEXT

This attribute indicates the degree of completion, or *status*, of a | Procedure |, as well as its various possible future states prior to its being initiated or completed.

Table 168: Permissible values for PROCEDURE CONTEXT

Attribute Values	Examples
Context values for actions 288532009 (<=)(< Q)	Operative procedure planned (situation) • ASSOCIATED PROCEDURE Surgical procedure (procedure) • PROCEDURE CONTEXT Planned (qualifier value)

6.2.2.9.9 TEMPORAL CONTEXT

This attribute indicates the *time* of occurrence of the situation, indicating whether the procedure or finding that it represents is actual and therefore occurred in the present, in the past, or at a specified *time*; or that it is planned or expected, that is, temporally located in the future. The most general value is simply | Current or past (actual) |, meaning that the *concept* was actual (not planned or expected), but not specifying anything further about its *time*. The word "specified" in the | TEMPORAL CONTEXT | values means that there is a date - *time* stamp associated with the code in the record, that gives a date and/or *time*, as a point and/or interval, that applies to the *concept*.

Table 169: Permissible values for TEMPORAL CONTEXT

Attribute Values	Examples
Temporal context value 410510008 (<=)(< Q)	History of - hematuria (situation) • ASSOCIATED FINDING Blood in urine (finding) • TEMPORAL CONTEXT In the past (qualifier value)

6.2.2.9.10 SUBJECT RELATIONSHIP CONTEXT

This attribute is used to specify the subject of the | Clinical finding | or | Procedure | being recorded, in relation to the subject of the record. In the example below, the subject of the record is the patient and the subject who smokes is the patient's father.

Table 170: Permissible values for SUBJECT Relationship CONTEXT

Concept Values	Examples
Person 125676002 (<=)(< Q)	Father smokes (situation) • ASSOCIATED FINDING Smoker (finding) • SUBJECT RELATIONSHIP CONTEXT Father of subject (person)

6.2.2.10 Attributes used to define Event concepts**Table 171: Approved Event attributes summary**

Defining Attribute	Subsumed Attribute	Allowable Values
ASSOCIATED WITH		Clinical Finding 404684003 (<<) Procedure 71388002 (<<) Event 272379006 (<<) Organism 410607006 (<<) Substance 105590001 (<<) Physical object 260787004 (<<) Physical force 78621006 (<<) Pharmaceutical / biologic product 373873005 (<< Q only) SNOMED CT Concept 138875005 (==)
	CAUSATIVE AGENT	Organism 410607006 (<<) Substance 105590001 (<<) Physical object 260787004 (<<) Physical force 78621006 (<<) Pharmaceutical / biologic product 373873005 (<< Q only) SNOMED CT Concept 138875005 (==)
	DUE TO	Clinical Finding 404684003 (<=) Event 272379006 (<=)
	AFTER	Clinical Finding 404684003 (<<) Procedure 71388002 (<<)
OCCURRENCE		Periods of life 282032007 (<)

 **Note:**

Meaning of Allowable Values (*Range*) notations:

- (<<) this code and *descendants*,
- (<) *descendants* only,
- (<=) *descendants* only (stated) except for supercategory groupers,
- (==) this code only,
- (< Q) *descendants* only when in a qualifying *Relationship*,
- (< Q only) *descendants* only, and only allowed in a qualifying *Relationship*.

For guidance and examples on the use of these attributes and value ranges to define events, see the section on clinical findings.

 **Note:** See also [Event](#).

6.2.2.11 Attributes used to define Physical Object concepts



Table 172: Approved Physical Object attributes summary

Defining Attribute	Allowable Values
HAS ACTIVE INGREDIENT	Substance 105590001 (<<)

 **Note:**

Allowable values for this attribute includes the *concept* listed and its *descendants*.

A limited number of *concepts* (e.g. drug-eluting stents) reside in the *Pharmaceutical/biologic product hierarchy* and the *Physical object hierarchy*. These *concepts* are all under | Drug-device combination product (product) |. This is the domain of | HAS ACTIVE INGREDIENT | within the *Physical Object hierarchy*. Editorial policies for the use of other attributes in the *Physical object hierarchy* generally, outside this particular domain, have yet to be established.

 **Note:** See also [Physical object](#).

6.2.2.12 Relationship groups in SNOMED CT



Multiple attributes and their values can be grouped together into “*Relationship groups*” to add clarity to *concept* definitions. A *Relationship group* combines an *attribute-value pair* with one or more other *attribute-value pairs*. *Relationship groups* originated to add clarity to | Clinical finding | *concepts* which require multiple | ASSOCIATED MORPHOLOGY | attributes and multiple | FINDING SITE | attributes and to | Procedure | which require multiple | METHOD | attributes and multiple | PROCEDURE SITE | attributes. However, *Relationship groups* are not limited to | Clinical finding | and | Procedure | *concepts*.

In the case of | Procedure |, *Relationship groups* generally associate the correct method with the correct site. In the example below, the *Relationship groups* clarify that there is exploration of the bile duct, and excision of the gall bladder. Without *Relationship groups*, the four attributes would be ungrouped and it would be unclear whether the excision was of the bile duct or of the gall bladder.

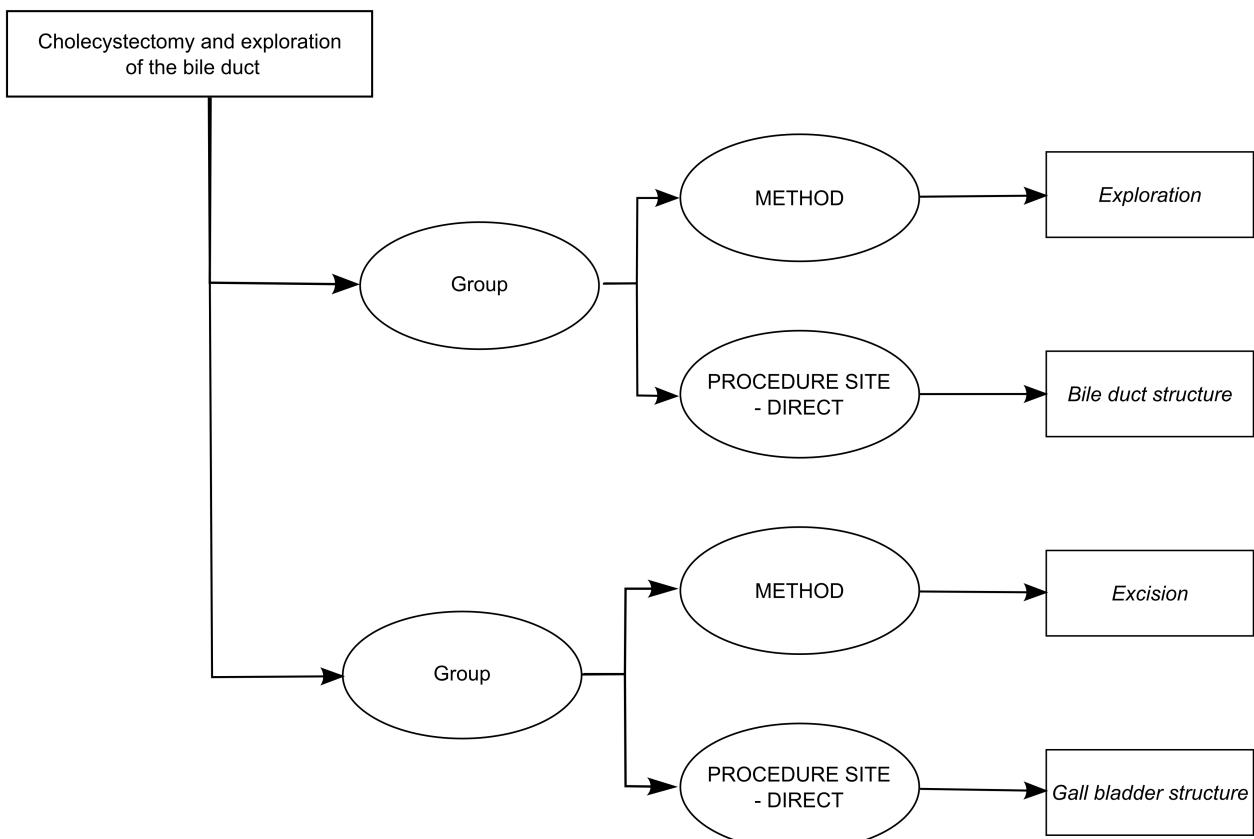


Figure 60: Example Cholecystectomy and exploration of bile duct

6.2.3 Miscellaneous Topics



6.2.3.1 References for Editorial Rules and Known Problems



The approved set of detailed editorial rules and guidelines are documented in the *SNOMED CT Editorial Guide*. Parts of the Editorial Guide are reproduced verbatim in the User Guide.

Known problems and issues are not documented here but instead are tracked on the *SNOMED CT Collaborative Space* at csfe.aceworkspace.net, under project "*IHTSDO*". It is possible to review a brief summary of each project without a login, but if you would like access to the *Collaborative Space*, please contact [collabnet\(at\)ihtsdo.org](mailto:collabnet(at)ihtsdo.org) with your contact details and a list of the project(s) to which you would like access. Known problems and issues are found in the content projects tracker under project "*IHTSDO*".

6.2.3.2 Terms Prefaced with Symbols



There are some *terms* in *SNOMED CT* that are prefaced with a symbol in square brackets.

These *concept codes* were inherited from *CTV3* and were used to facilitate mapping to *ICD-10*. They have all been *retired* by moving them to the *UK NHS extension*, and are not recommended for use in clinical records.

Explanations of these *term* prefixes are as follows:

Table 173: Term Preface Symbols

[X]	Terms starting with [X] were initially used in the <i>Read Codes</i> in the 1995 release, in order to identify <i>ICD-10 terms</i> that were not present in <i>ICD-9</i> .
-----	--

- [D]** Terms starting with [D] are also from CTV3, and identify terms contained in ICD-9 Chapter XVI 'Symptoms signs and ill-defined conditions' and ICD-10 Chapter XVIII 'Symptoms signs and abnormal clinical and laboratory findings, not elsewhere classified'. The [D] meant that in CTV3 the code was intended for use in a diagnosis field in the record, even though the term meaning is not a kind of disease.
- [V]** A term starting with [V] identifies concept codes derived from ICD-9 'Supplementary classification of factors influencing health status and contact with health services (V codes)', and ICD-10 Chapter XXI 'Factors influencing health status and contact with health services (Z codes)'.
- [M]** A term starting with [M] identifies Morphology of Neoplasm terms present in ICD-9 and ICD 10.
- [SO]** A term starting with [SO] signifies that the term was contained in OPCS-4 (Office of Population, Censuses and Surveys - Classification of Surgical Operations and Procedures - 4th Revision) Chapter Z subsidiary classification of sites of operation in CTV3.
- [Q]** A term starting with [Q] identifies temporary qualifying terms inherited from CTV3.

6.2.3.3 Negation



The meaning of some concept codes in SNOMED CT depends conceptually on negation (e.g. absence of X, lack of X, unable to do X etc).

6.2.3.3.1 Negation and Context



The | Situation with explicit context | hierarchy is intended to manage this kind of semantic situation. The concept model allows a concept code in the | Situation with explicit context | hierarchy to be related to the | Clinical finding | about which context is asserted. For example, | Absence of nausea and vomiting (situation) | is modeled as a | Situation with explicit context | in which the finding of | Nausea and vomiting (disorder) | is absent.

The inclusion of negated meanings introduces complications into query formulation, machine classification, and reasoning tasks. The inclusion of a NOT logical operator into the SNOMED CT compositional model could simplify modeling of negated meanings. The current release of SNOMED CT does not directly support classification using this operator, but some modeling formalisms in current use today (including database formalisms, Description Logic formalisms) include a NOT operator as a fundamental modeling primitive.

6.3 Machine Readable Concept Model



The Concept Model is the set of rules that govern the ways in which SNOMED CT concepts are permitted to be modeled using Relationships to other Concepts. The Machine Readable Concept Model (MRCM) represents these rules in a form that can be read by a computer and applied to test that concept definitions comply with the rules.

The primary requirements addressed by the current MRCM relate to supporting content authoring and validation prior to distribution. However, the MRCM also has a potential value for implementers as a source of the rules that determine whether particular postcoordinated expression refinements are permitted.

The MRCM is based on a logical model that specifies:

- CM-Domains: Sets of SNOMED CT concepts to which a common set of Concept Model Constraints apply.
- CM-Attributes: Sets of SNOMED CT concepts that can be used as relationship types.
- CM-Ranges: Sets of SNOMED CT concepts that can be used as values for a particular defining Relationship.

- CM-Constraints: Rules that determine which combinations of CM-Attributes and CM-Ranges may be applied to *concepts* in a CM-Domain.

The current prototype version of the *MRCM* is represented as a relational database schema with an XML schema to support export and import of data.

Subsequent activities within *IHTSDO* Working Group and related work by *IHTSDO Members* has identified requirements for additional representations that are more readily refinable to support implementation use-cases.

 **Note:** Future releases of this guide will provide additional information advice on practical uses of the *MRCM*. In the meantime, documentation about the *MRCM* and *MRCM* export files are available separately from the *IHTSDO*.

Chapter

7

7 Terminology Services Guide



7.1 Representing SNOMED CT resources



7.1.1 Choosing a terminology server view



SNOMED CT Release Format 2 is designed to enable the distribution and use of a full historical view of *SNOMED CT* from its first release in 2002 up to its most recent release. This allows *terminology servers* to provide a range of different views of *SNOMED CT*. However, it does not require that all *terminology servers* support the full range of views.

Table 174 identifies three options for the views that a *SNOMED CT terminology server* may support. The simplest of these is the single *snapshot view* which provides access to a single *release version*. This closely matches the view provided by the original *SNOMED CT release format (RF1)*. The most powerful *full view* which allows the server to provide access to any selected version of *SNOMED CT* from a single representation of the *SNOMED CT* resource. This makes full use of the version features in *RF2*. Alternatively a server may provide a selected set of snapshots representing versions of known interest to its users.

People designing a *terminology server* need to decide whether their server will only provide access to a single current view of the *SNOMED CT* resource or will also support retrospective views of earlier versions of the terminology. The single *snapshot view* is simplest to implement and matches the service most vendors offered with original *SNOMED CT release format (RF1)*. A more complete view is now possible using *Release format 2* and this offers several significant advantages. It supports incremental updates allowing smoother transition as new versions become available. It also allows changes between versions to be detected more easily and can be used to evaluate queries against an earlier version for comparative purposes.

People choosing a *terminology server* need to consider whether a server that only supports a single *snapshot view* of the current version meets their requirements. If they require access to previous versions a server that supports the *full view* is likely to be the best long term solution. A server that allows access to multiple discrete snapshots may provide a reasonable interim solution but may be less flexible and less easy to maintain.

Table 174: SNOMED CT views that may be supported by terminology servers

View	Description
<i>Snapshot view</i>	A <i>snapshot view terminology service</i> provides access to the content of the current state of all the <i>components</i> of the <i>International Release</i> and any chosen <i>Extension Releases</i> .

View	Description
<p><i>Multi-snapshot view</i></p>	<p>A "multi-snapshot view" terminology service provides access to:</p> <ul style="list-style-type: none"> • the content of the current state and content of all <i>components</i> of the <i>International release</i> and any chosen <i>Extension Releases</i>; • the content of one or more additional <i>snapshot views</i>, each of which represents the state of all <i>components</i> at a different fixed point in time. <p>A "multi-snapshot view" terminology server may provide access to delta views that report the differences between two <i>snapshot views</i>. This is limited to comparisons of specific points represented by the available <i>snapshot views</i>.</p>
<p><i>Full view</i></p>	<p>A <i>Full view terminology service</i> provides access to:</p> <ul style="list-style-type: none"> • the complete content of the full <i>International release</i> and any chosen <i>Extension Releases</i>; • the state and content of all <i>components</i> as they were at any specified point in time. <p>A <i>full view terminology server</i> should also provide access to views that show the changes to <i>components</i> between any two specified points in time.</p>

The *full view* is required to support some *SNOMED CT* use cases but many requirements can be adequately met by providing access to a current *Snapshot view*. The *multi-snapshot view* is an approach that may meet some requirements that are not met by a single snapshot without requiring support for the *Full view*.

 **Note:** terminology servers that do not support the *Full view* still need to be able to import from a *Full release* as *Extension providers* are not required to provide the *snapshot* or *delta releases* ([Importing release types](#))

7.1.2 Choosing a technical approach



People designing a *terminology server* need to decide how they will store and access the *SNOMED CT* resources. This decision depends on a variety of factors including: types of *Terminology services* required, the technical environment in which development is undertaken and the experience of the developers.

People choosing a *terminology server* need to know whether the server will meet their requirements and whether it works effectively in their preferred technical environment. They will also wish to be sure it delivers the required functionality and performance. While they may not be directly interested in technical approach to representation of *SNOMED CT* resources, these design decisions are likely to affect the ability of a server to meet their requirements.

The following sub-sections briefly outline some of the technical options.

7.1.2.1 Direct use of release files in a relational database



The distributed *release files* can be imported directly into a database schema that matches the distribution file specification. This data then provides the core resource at the heart of a *terminology server*.

This direct use of distributed files in a relational database has the advantage of allowing simple installation. However, it may not be the most efficient approach in terms of performance or file size. Some *terminology services* require relatively complex queries with multiple joins, and need to be completed in fractions of a second to provide an acceptable *user interface*.

👉 **Example:** To display the set of *subtype children* of a *concept* with their *preferred terms* in a specified language or *dialect* requires joins several joins between *concepts*, *Relationships*, *Descriptions* and a *language refset*.

To search for a term matching a supplied pattern in a *concept* that represents a type of procedure also requires multiple joins to link the *Descriptions* with matching terms to the relevant *concept* and test whether it is a *subtype* of the 71388002 | Procedure (procedure) | *concept*.

The performance criteria of searches and joins in very large relational databases vary significantly. Therefore, different optimizations may need to be used to achieve acceptable response times according to the nature of the relational database system.

An additional consideration for *RF2* implementations is the way in which alternative views are supported since, without optimization these may have a significant impact on performance.

7.1.2.2 Alternative relational structures



There is no requirement to use the data structure as distributed. Other structures can be used provided that they are able to deliver the range of *Terminology services* required. Options include:

- Partially denormalized representations that omit direct representation of some components.

👉 **Example:** Frequently used information distributed as part of a *Refset* could be represented by direct inclusion of the added information as additional columns in the table representing the referenced component.
- Omission of some of the tables where a particular function is not required.

👉 **Example:** The *Refset* tables representing *cross maps* could be omitted if the intended uses of the *terminology server* explicitly exclude *cross mapping*.
- Replacement of some of the supporting tables with proprietary alternatives that deliver equivalent or enhanced functionality.

👉 **Example:** The word search support tables could be replaced by other tables or indices generated by the *terminology server* when loading the distribution files.

7.1.2.3 Non-relational structures



Although the primary distribution format is relational, this does not require *terminology servers* to utilize a relational database as the primary or only storage format. The requirements for *terminology services* may also be met by representing some or all of the distributed data in other forms including object-oriented databases, Extensible Mark-up Language (XML) and/or proprietary data structures. These structures may be used separately or, in some cases, in combination with a relational database.

7.1.3 Example of a Full View Relational Representation



This section outlines an example of a relational approach to representation of a *full view* of the *SNOMED CT* Resources. The example has been developed and tested using the Open Source database MySql Community Edition.

The example schema is based closely on the *RF2* structure and is used in subsequent discussions of implementation issue and options for addressing those issues.

 **Note:** The approach described here is only an illustrative example. It shows one way to represent the data but should not be interpreted as a recommended or standard approach.

The general approach is as follows:

- Each datatype in the *RF2* specification is expressed with a common mapping to a database datatype:
 - Alternative implementations following the same general pattern could use a different datatype map but the mapping should be consistent within an implementation. Reasons for different datatype maps include implementer preferences and the capabilities of the database.
- Each of the main file types specified in *RF2* is instantiated as a database table:
 - Each table is named for the *component* type (e.g. *sct2_Concept*, *sct2_Description*, *sct2_Relationship*, *sct2_Identifier*).
 - Each field in these tables has column name from the *release file*
 - Each field is assigned the appropriate datatype (and where appropriate size).
- *Refsets* are represented slightly differently from the other files:
 - One table structure for each distinct structure present in the release data:
 - *der2_Refset*.
 - *der2_Reset_c*.
 - *der2_Refset_cc*.
 - *der2_Refset_ci*.
 - *der2_Refset_i*.
 - *der2_Refset_s*.
 - *der2_Refset_ss*.
 - ... etc as new structures are added.
 - The first six fields in these tables have the common column names from the *release file*
 - The subsequent fields are named by type and position:
 - *sctid1*.
 - *string1*.
 - *integer1*.
 - ... etc.
 - This polymorphic field approach to column naming is used because column names may vary between *release files* for different *Reference Set* patterns, even when column data types are the same.

 **Note:** Two other approaches could be used here.

1. A separate table for each type of *Refset* based on column names rather than on structure. This would require several tables with similar types of *Relationships* to other *components*.
2. A single general purpose *Refset* table with multiple polymorphic fields. For example, *strings* that could be used to represent the other data types. This could cause inefficiencies for *sctid* type fields as the joins between these and target *components* would be heterogeneous.

7.1.3.1 Example Datatype Mapping for Relational View



The following table provides example mapping from the *SNOMED CT RF2* datatypes to appropriate datatypes supported by MySQL.

Table 175: Example Datatype Mappings

RF2 Datatype	MySQL Datatype	Comment on Mapping
<i>SCTID</i>	BIGINT	Both these datatypes represent 64-bit integers.
<i>UUID</i>	BINARY(16)	MySQL does not have a native datatype for <i>UUID</i> . The BINARY(16) representation is most economical for storage and most efficient for indexing. This requires a transformation on storage or review. The example queries in this guide use the simple transformations functions shown in Table 176 . An alternative is to use CHAR or VARCHAR representations. This does not require the transformations noted above. However, use of VARCHAR (36) costs 38 bytes rather than 16 bytes per <i>UUID</i> and due to use of UTF8 using CHAR (36) consumes a fixed 108 bytes per UID in a MySQL table. More importantly the index performance is poorer for these <i>string</i> representations.
<i>Integer</i>	INT	Both these datatypes represent 32-bit integers.
<i>String</i>	VARCHAR (Len)	VARCHAR is used in preference to CHAR as it provides more space efficient storage. Note that in the UTF8 encoded tables required for the MyISAM database reserves three bytes per character for fixed length <i>strings</i> . In contrast VARCHAR uses the number of bytes actually plus one or two bytes to specify length. Use of VARCHAR does result in some loss of performance but <i>strings</i> are only used in <i>Descriptions</i> , <i>string refs</i> and <i>Identifier</i> tables. In all these cases <i>strings</i> with a significant range of lengths are used and the space penalty for using CHAR datatypes would be high.
<i>Boolean</i>	TINYINT	MySQL treats the datatype name <i>boolean</i> as an alias for TINYINT. In the examples this mapping is made explicit.
<i>Time</i>	DATETIME	This is the full representation of date and time and is used to ensure compatibility with existing data and potential accommodation of time stamped data. The more compact DATE type could be used with current data as the <i>effectiveTime</i> is currently a date only representation. However, the more flexible DATETIME has been preferred in the examples because this emphasizes the fact that in an International environment the <i>effectiveTime</i> implies the UTC time and thus the date alone is not a precise representation.

Table 176: Example UUID transformation

Action	
Load or insert to storage	SET [column-name] = UNHEX(REPLACE(@uid,'-',''))
Select from storage	RenderUid([column-name])

Action	
UNHEX	A built in MySql function that converts a hexadecimal <i>string</i> to binary.
RenderUid	FUNCTION `RenderUid`(Uid blob) RETURNS varchar(36) CHARSET utf8 BEGIN Set @Tmp = Hex(uid); RETURN CONCAT(SUBSTRING(@Tmp,1,8),'-',SUBSTRING(@Tmp,9,4),'-', SUBSTRING(@Tmp,13,4),'-',SUBSTRING(@Tmp,17,4),'-',SUBSTRING(@Tmp,21)); END

7.1.3.2 Example Full View Concept Table



```
CREATE TABLE `sct2_concept` (
`id` BIGINT NOT NULL DEFAULT 0,
`effectiveTime` DATETIME NOT NULL DEFAULT '0000-00-00 00:00:00',
`active` TINYINT NOT NULL DEFAULT 0,
`moduleId` BIGINT NOT NULL DEFAULT 0,
`definitionStatusId` BIGINT NOT NULL DEFAULT 0,
PRIMARY KEY (`id`, `effectiveTime`)
) ENGINE=MyISAM DEFAULT CHARSET=utf8;
```

Figure 61: Create Concept Table

👉 **Tip:** Some of the approaches to optimization suggested elsewhere in the guide result in changes to this example schema. You may wish to consider these before implementing this schema.

```
LOAD DATA LOCAL INFILE '[path]sct2_concept_[AdditionalInfo].txt'
INTO TABLE `sct2_concept`
LINES TERMINATED BY '\r\n' IGNORE 1 LINES;
```

Figure 62: Import Concept file

7.1.3.3 Example Full View Description Table



```
CREATE TABLE `sct2_description` (
`id` BIGINT NOT NULL DEFAULT 0,
`effectiveTime` DATETIME NOT NULL DEFAULT '0000-00-00 00:00:00',
`active` TINYINT NOT NULL DEFAULT 0,
`moduleId` BIGINT NOT NULL DEFAULT 0,
`conceptId` BIGINT NOT NULL DEFAULT 0,
`languageCode` VARCHAR(3) NOT NULL DEFAULT '',
`typeId` BIGINT NOT NULL DEFAULT 0,
`Term` VARCHAR(255) NOT NULL DEFAULT '',
`caseSignificanceId` BIGINT NOT NULL DEFAULT 0,
PRIMARY KEY (`id`, `effectiveTime`),
KEY `sct2_description_concept` (`conceptId`)
) ENGINE=MyISAM DEFAULT CHARSET=utf8
```

Figure 63: Create Description Table

 **Tip:** Some of the approaches to optimization suggested elsewhere in the guide result in changes to this example schema. You may wish to consider these before implementing this schema.

```
LOAD DATA LOCAL INFILE '[path]sct2_description_[AdditionalInfo].txt'
INTO TABLE sct2_description
LINES TERMINATED BY '\r\n' IGNORE 1 LINES;
```

Figure 64: Import Description file

```
CREATE INDEX ix_sct2_description_3 ON sct2_description([ConceptId],[typeId],[languageCode])
```

Figure 65: Index Description Table - Concept

7.1.3.4 Example Full View relationships table



```
CREATE TABLE `sct2_relationship` (
  `id` BIGINT NOT NULL DEFAULT 0,
  `effectiveTime` DATETIME NOT NULL DEFAULT '0000-00-00 00:00:00',
  `active` TINYINT NOT NULL DEFAULT 0,
  `moduleId` BIGINT NOT NULL DEFAULT 0,
  `sourceId` BIGINT NOT NULL DEFAULT 0,
  `destinationId` BIGINT NOT NULL DEFAULT 0,
  `relationshipGroup` INT NOT NULL DEFAULT 0,
  `typeId` BIGINT NOT NULL DEFAULT 0,
  `characteristicTypeId` BIGINT NOT NULL DEFAULT 0,
  `modifierId` BIGINT NOT NULL DEFAULT 0,
  PRIMARY KEY (`id`, `effectiveTime`),
  KEY `sct2_relationship_source` (`sourceId`, `characteristicTypeId`, `typeId`, `destinationId`),
  KEY `sct2_relationship_dest` (`destinationId`, `characteristicTypeId`, `typeId`)
) ENGINE=MyISAM DEFAULT CHARSET=utf8
```

Figure 66: Create relationships table

 **Tip:** Some of the approaches to optimization suggested elsewhere in the guide result in changes to this example schema. You may wish to consider these before implementing this schema.

```
LOAD DATA LOCAL INFILE '[path]sct2_relationship_[AdditionalInfo].txt'
INTO TABLE sct2_relationship
LINES TERMINATED BY '\r\n' IGNORE 1 LINES;
```

Figure 67: Import Relationship file

7.1.3.5 Example Full View Identifier Table



```
CREATE TABLE `sct2_identifier` (
  `identifierSchemaId` BIGINT NOT NULL DEFAULT 0,
  `alternateIdentifier` VARCHAR(255) NOT NULL DEFAULT '',
  `effectiveTime` DATETIME NOT NULL DEFAULT '0000-00-00 00:00:00',
  `active` TINYINT NOT NULL DEFAULT 0,
  `moduleId` BIGINT NOT NULL DEFAULT 0,
  `referencedComponentId` BIGINT NOT NULL DEFAULT 0,
  PRIMARY KEY (`identifierSchemaId`, `alternateIdentifier`, `effectiveTime`),
  KEY `sct2_relationship_sctid` (`referencedComponentId`)
) ENGINE=MyISAM DEFAULT CHARSET=utf8
```

Figure 68: Create Identifier Table

 **Tip:** Some of the approaches to optimization suggested elsewhere in the guide result in changes to this example schema. You may wish to consider these before implementing this schema.

```
LOAD DATA LOCAL INFILE '[path]sct2_identifier_[AdditionalInfo].txt'
INTO TABLE sct2_identifier
LINES TERMINATED BY '\r\n' IGNORE 1 LINES;
```

Figure 69: Index Identifier Table - Primary

7.1.3.6 Example Full View Refset Table



```
CREATE TABLE `sct2_refset_c` (
  `id` binary(16) NOT NULL DEFAULT '0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0',
  `effectiveTime` DATETIME NOT NULL DEFAULT '0000-00-00 00:00:00',
  `active` TINYINT NOT NULL DEFAULT 0,
  `moduleId` BIGINT NOT NULL DEFAULT 0,
  `refsetId` BIGINT NOT NULL DEFAULT 0,
  `referencedComponentId` BIGINT NOT NULL DEFAULT 0,
  `sctId1` BIGINT NOT NULL DEFAULT 0,
  PRIMARY KEY (`id`, `effectiveTime`),
  KEY `refset_c_id` (`refsetId`, `referencedComponentId`)
) ENGINE=MyISAM DEFAULT CHARSET=utf8;
```

Figure 70: Create Component Refset Table

 **Tip:** Some of the approaches to optimization suggested elsewhere in the guide result in changes to this example schema. You may wish to consider these before implementing this schema.

```
LOAD DATA LOCAL INFILE '[path]sct_cRefset_[AdditionalInfo].txt'
INTO TABLE `sct2_refset_c`
LINES TERMINATED BY '\r\n' IGNORE 1 LINES
(@uid, `effectiveTime`, `active`, `moduleId`, `refsetId`, `referencedComponentId`, `sctId1`)
SET id=UNHEX(REPLACE(@uid,'-',''));
```

Figure 71: Import Component Refset File

7.2 Importing SNOMED CT release data



7.2.1 Choosing a Release Type to import



The first step in selecting the set of *release files* to be imported is to decide which *Release Type* will be used. *SNOMED CT Release Format 2* specifies three distinct *Release Types*: *full release*, *snapshot release* and *delta release*. These are described in the table below.

The *Release Format 2* specification states that:

- The *SNOMED CT International Release* will include all three *Release Types*;
- A *SNOMED CT Extension Release* must include the *full release*;
- A *SNOMED CT Extension Release* may optionally include a *snapshot release* and/or *delta release*.

A *SNOMED CT-enabled terminology server* must be able to import data from a *full release* because this is the only *Release Type* that is required to be produced by all *Extension* developers. A *SNOMED CT-enabled terminology server* should also be able to import from other *Release Types* where these are available as these may allow more efficient updating.

The choice of a particular *Release Type* depends on the type of *terminology views* that the *terminology server* is designed to support and on whether this is an initial import or a subsequent update.

 **Note:** The requirement to be able to import data from the *full release* does not mean that all *terminology servers* must provide access to the complete historical set of data provided by a *full release*. The *full release* can be selectively imported to used to populate a *snapshot view* for applications that do not require access to historical data.

7.2.1.1 Release Types



[Table 177](#) specifies the content of each of the *Release Format 2 Release Types*.

This table is followed by illustrations of each of the *Release Types* using the small same pattern of content development over seven release cycles. These illustrations highlight the key differences and the *Relationships* between the *Release Types*.

Table 177: SNOMED CT Release Types

Release Type	Description
Full	The files representing each type of component contain every version of every component ever released.
Snapshot	The files representing each type of component contain one version of every component released up to the time of the snapshot. The version of each component contained in a snapshot is the most recent version of that component at the time of the snapshot.
Delta	The files representing each type of component contain only component versions created since the previous release. Each component version in a <i>delta release</i> represents either a new component or a change to an existing component.

The seven columns in each of the following illustrations represent the content of seven releases (numbered 1-7). Each *component* is identified by a letters (A-K). A *component* version is represented by the identifying letter followed by a number (1-7) representing the release cycle in which that *component* version became effective.

[Figure 72](#) shows the content of a series of *full releases*. The yellow background color highlights the set of *component* versions that are also present in the snapshot for the same *release version* (see [Figure 74](#)). *component* versions are shown in gray in releases versions after they have been superseded by a new *component* version. Newly added *component* versions, shown in red, are also present in the delta for the same *release version* (see [Figure 73](#)).

The content of the *full release* in any chosen version is identical to the combined content of all the *snapshot releases* up to and including that version. Thus adding a *delta release* to the previous version of the *full release* creates the *full release* for the new version. The *snapshot release* is derived from the *full release* by removing all except the most recent version of each *component*.

Earlier <-- FULL RELEASES FOR SEVEN RELEASE CYCLES --> Later						
1	2	3	4	5	6	7
A,1	A,1	A,1	A,1	A,1	A,1	A,1
B,1	B,1	B,1	A,4	A,4	A,4	A,4
C,1	B,2	B,2	B,1	B,1	B,1	B,1
D,1	C,1	C,1	B,2	B,2	B,2	B,2
E,1	D,1	C,3	C,1	C,1	B,6	B,6
	E,1	D,1	C,3	C,3	C,1	C,1
F,2	E,1	C,4	C,4	C,3	C,3	C,3
G,2	F,2	D,1	D,1	C,4	C,4	C,4
	F,3	E,1	E,1	C,6	C,6	C,6
	G,2	F,2	F,2	D,1	D,1	D,1
	H,3	F,3	F,3	E,1	E,1	E,1
		G,2	G,2	F,2	F,2	F,2
		H,3	H,3	F,3	F,3	F,3
		I,4	I,4	G,2	G,2	G,2
			J,5	H,3	H,3	H,3
				H,6	H,6	H,6
				I,4	H,7	H,7
				J,5	I,4	I,4
					J,5	J,5
						K,7

Figure 72: Full release illustration

Earlier<-- DELTA RELEASES FOR SEVEN RELEASE CYCLES --> Later						
1	2	3	4	5	6	7
A,1			A,4			
B,1	B,2				B,6	
C,1		C,3	C,4		C,6	
D,1						
E,1		F,2	F,3			
		G,2	H,3		H,6	H,7
			I,4		J,5	
						K,7

Figure 73: Delta release illustration

Earlier<-- SNAPSHOT RELEASES FOR SEVEN RELEASE CYCLES -->Later						
1	2	3	4	5	6	7
A,1	A,1	A,1	A,4	A,4	A,4	A,4
B,1	B,2	B,2	B,2	B,2	B,6	B,6
C,1	C,1	C,3	C,4	C,4	C,6	C,6
D,1	D,1	D,1	D,1	D,1	D,1	D,1
E,1	E,1	E,1	E,1	E,1	E,1	E,1
	F,2	F,3	F,3	F,3	F,3	F,3
	G,2	G,2	G,2	G,2	G,2	G,2
		H,3	H,3	H,3	H,6	H,7
			I,4	I,4	I,4	I,4
				J,5	J,5	J,5
						K,7

Figure 74: Snapshot release illustration

👉 Note: In a real SNOMED CT release each of the letters A-K would be replaced by a *component id* (a *SNOMED CT identifier*) and each of the release cycle numbers 1-7 would be replaced by the *effectiveTime* of a *release version*.

7.2.1.2 Importing and maintaining a Full view

7.2.1.2.1 Importing a Full view



To provide access to the *full view* of the content of the *SNOMED CT International Release*, a terminology server must initially import content from the *full release files* for the *International Release*.

The complete content of all the main *release files* should be imported into the chosen internal representation.

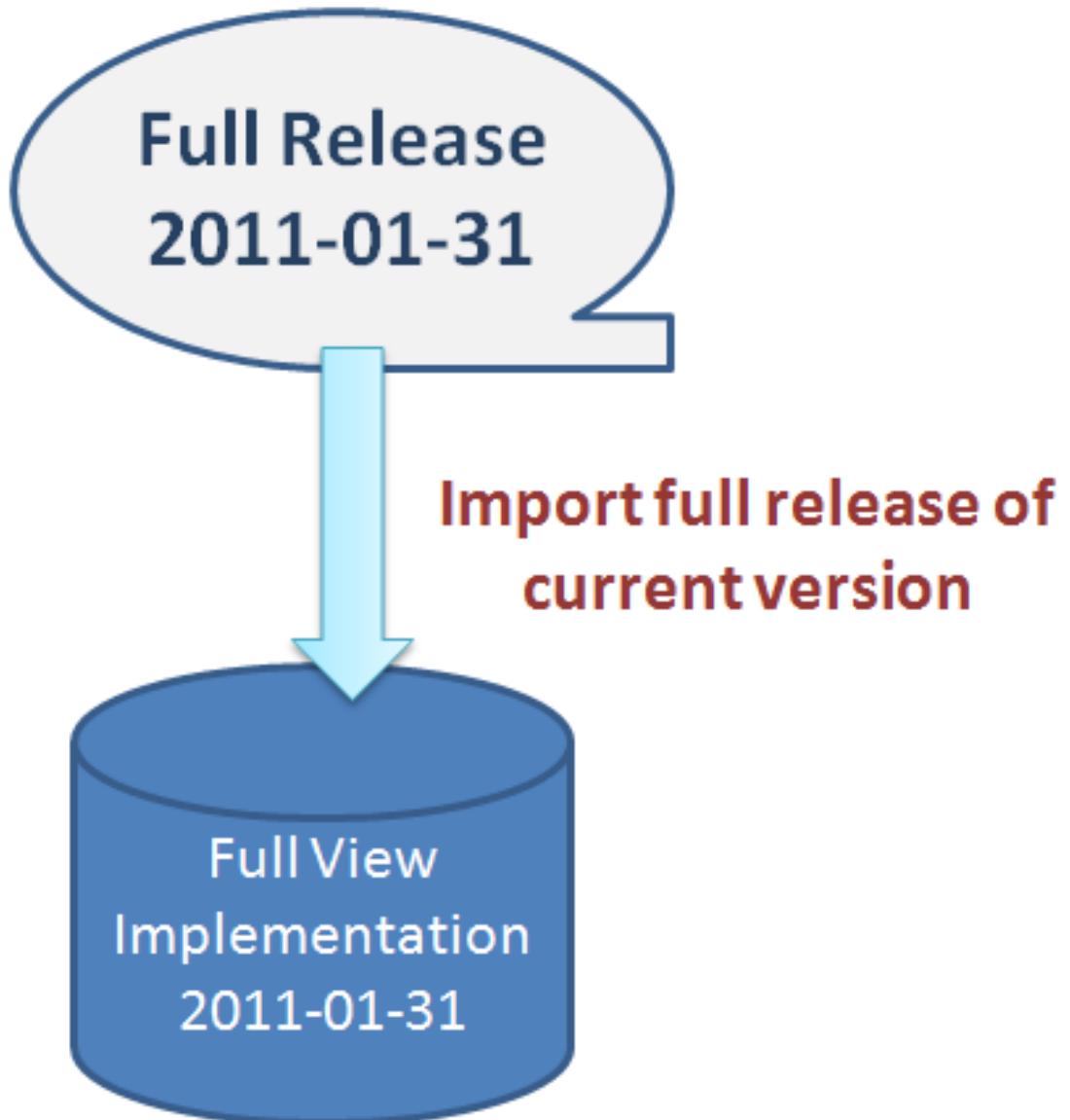


Figure 75: Initial import to create a full view

 **Tip:** The files that form part of a particular release can be identified by pattern matching based on the IHTSDO filenames (see [Identifying release files using regular expressions](#)).

7.2.1.2.2 Updating a Full view



A *full view* can be updated by one of the following approaches:

1. Append the content of the relevant *delta release* files to a previously created *full view*:

- The delta files contain only the changes since a previous release. Appending the data from these files to the *full view* for the previous version creates the *full view* for the new version. There is no need to change or delete existing data.

Caution: A *delta release* must be applied to the immediately previous version. Appending a *delta release* to earlier versions will result in omission of content and this will lead to significant errors when interpreting the data.

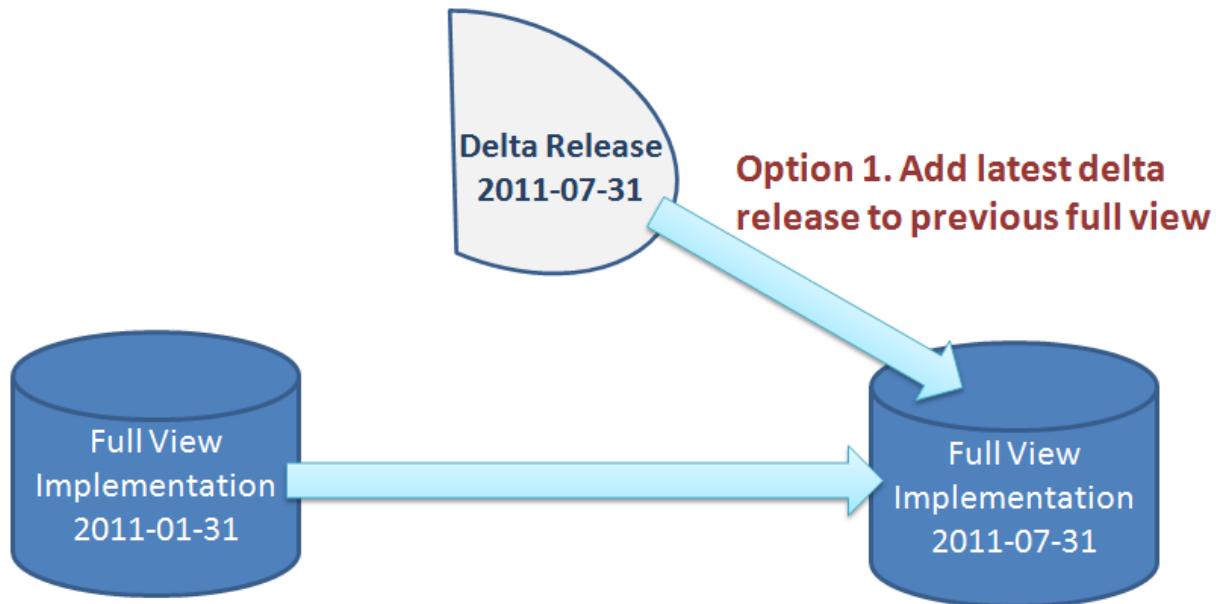


Figure 76: Updating a full view using a snapshot release

2. Filter the relevant *full release* files to generate a *delta release* then apply this as in 2 above:

- The *delta release* consists of all items in the *full release* with an *effectiveTime* greater than the *effectiveTime* of the most recent previous release. Therefore, it is easy to filter a *full release* to generate a set of *delta release* files.
 - Alternatively a "virtual delta" release may be used by filtering the *full release* while importing.
- Note:** This allows *Extensions* that are not distributed with *delta releases* to be processed by a general update process that is optimized to work with *delta releases*.

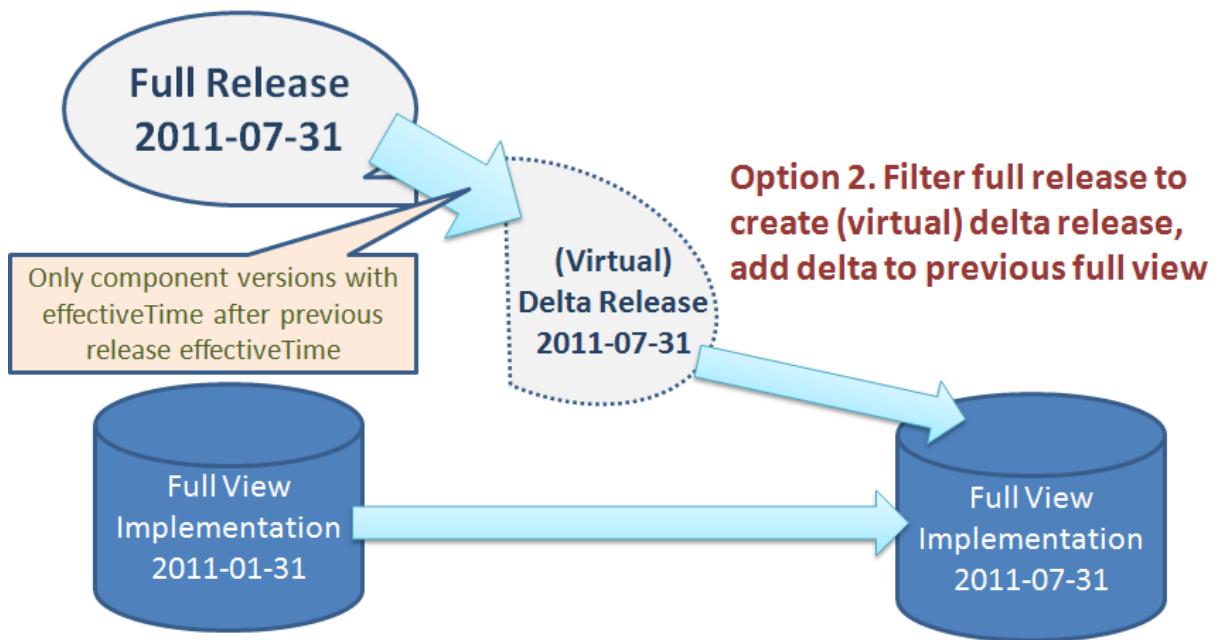


Figure 77: Updating a full view using a full release

3. Use the *full release* files to completely replace previously imported data:
 - This follows the approach described earlier to [import a full view](#).

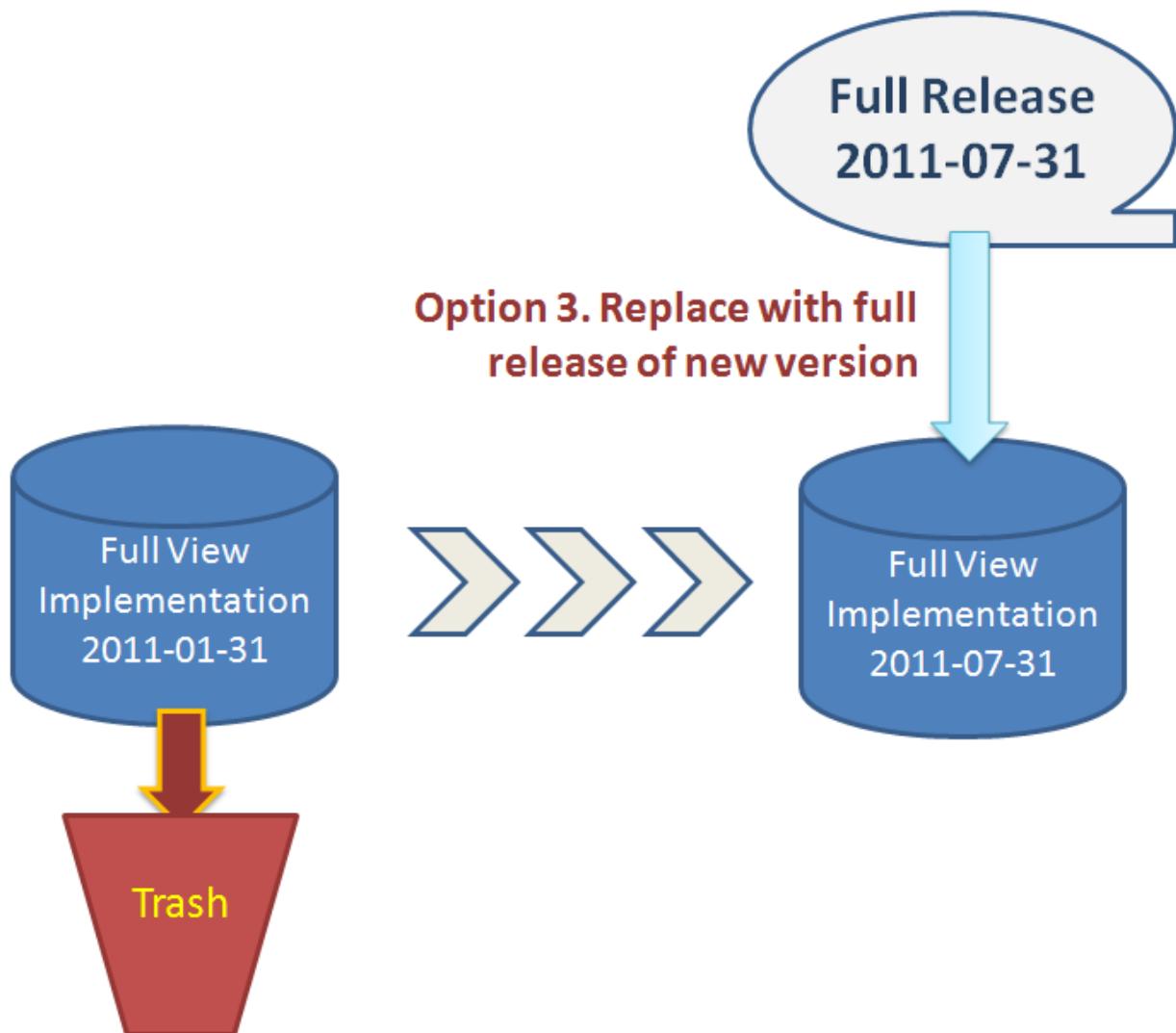


Figure 78: Updating a full view by replacement

👉 **Tip:** The files that form part of a particular release can be identified by pattern matching based on the IHTSDO filenaming conventions (see [Identifying release files using regular expressions](#)).

7.2.1.2.3 Importing and updating Extensions for a Full view



To provide access to the *full view* of one or more *Extensions*, a *terminology server* must initially import content from the *full release* files for each of the required *Extensions*. Thereafter, the *full view* of each *Extension* can be maintained by using any of the techniques described for [updating a full view](#).

When a *full view* of *Extension* data is initially imported or subsequently updated care needs to be taken to ensure the relevant versions of the *International Release* and any other *Extensions* on which it depends have been imported. Failure to follow do this may lead to errors as a result of references from *Extension components* to missing or out of date *components* in the *International Release* or in another *Extension*.

A *full view* may include more recent versions of the *International Release* than is required to support the *Extension*. In this case, when the *Extension* is viewed the *International Release* can, if necessary, be viewed as it would have been in the version to which the *Extension* is related. Similarly, if one *Extension* depends on content in another *Extension*, the version the *Extension* on which it depends may be a more recent version.

The table below summarizes the compatibility between the *full views* of given versions of an *Extension* and the *International Release*. It also indicates the ways in which a *full view* may be used when the latest installed

versions are not directly compatible. If one *Extension* depends on another *Extension*, the same considerations apply to compatibility between the versions of those *Extensions*.

Table 178: Compatibility between full views of versions of an Extension and the International Release

Relationship between the version of the <i>Extension</i> and the <i>International Edition</i>	Notes on compatibility and usability
Installed <i>International Release</i> is older than the version on which the <i>Extension</i> was based	<p>Incompatible - unless recent <i>Extension</i> content is excluded.</p> <p>The <i>Extension</i> may include <i>Relationships</i> to <i>concepts</i> that do not exist in this version of the <i>International Release</i>. This will lead to errors that cannot be reconciled while viewing the <i>Extension</i> content.</p> <p>A system with this mix of installed versions could be safely used by excluding the content of the more recent <i>Extension</i> versions. This can be done by excluding any <i>Extension component-version</i> with an <i>effectiveTime</i> of one of the versions based on a newer <i>International Release</i>. In effect this approach rolls back the <i>Extension</i> to the last <i>Extension</i> that is valid with the installed version of the <i>International Release</i>.</p>
Installed <i>International Release</i> is same version as the one on which the <i>Extension</i> was based	<p>Fully compatible.</p> <p>This is the version the <i>Extension</i> was created for so it should behave as intended.</p>

Relationship between the version of the Extension and the International Edition	Notes on compatibility and usability
<p>Installed <i>International Release</i> is newer than the version on which the <i>Extension</i> was based.</p>	<p>Compatible - subject to appropriate configuration and usage.</p> <p>The <i>International Edition</i> for this version may include:</p> <ul style="list-style-type: none"> Additional <i>components</i>. These will not cause errors because the <i>International Release</i> does not reference the <i>Extension</i> and the <i>Extension</i> content cannot reference <i>components</i> that did not exist when in the version it was based on. Changes to the state of some <i>components</i>. These changes may affect the interpretation of some parts of the <i>Extension</i>. <p>However, despite these issues the <i>full view</i> resulting from this combination can be used in several ways:</p> <ol style="list-style-type: none"> Configured to roll-back the <i>International Release</i> to the version on which the <i>Extension</i> was based. This can be done with a <i>full release</i> by creating a virtual view of the <i>International Release components</i> which excludes <i>component</i> versions with an <i>effectiveTime</i> greater than the version on which the <i>Extension</i> was based. This type of view is described in more detail in Implementing the State-Valid view. Configured to exclude the <i>Extension</i>. In this case the most recent version of the <i>International Release</i> can be viewed. Configured to use those parts of the <i>Extension</i> that support translation of <i>International Release</i> content. In this case, the <i>Extension</i> will enable translated rendering of pre-existing translated content. This would leave new and untranslated <i>concepts</i> to be rendered in English (or another available language). Accepting and working within the constraints imposed by the omissions and anomalies noted above. This mode should not be used routinely but may be useful for assessing the impact of changes to the <i>International Release</i> on the <i>Extension</i>.

 **Note:** The compatibility and usability notes are specific to a *full view* implementation. Different considerations apply to *snapshot* and multi-*snapshot* views.

7.2.1.3 Importing and maintaining a Snapshot view

7.2.1.3.1 Importing a Snapshot view



To provide access to the *snapshot view* of the content of the *SNOMED CT International Release*, a *terminology server* must initially do one of the following:

- Import the content from a set of *snapshot release* files for a version of the *International Release*:
 - All the rows from a set of *snapshot release* files must be imported.

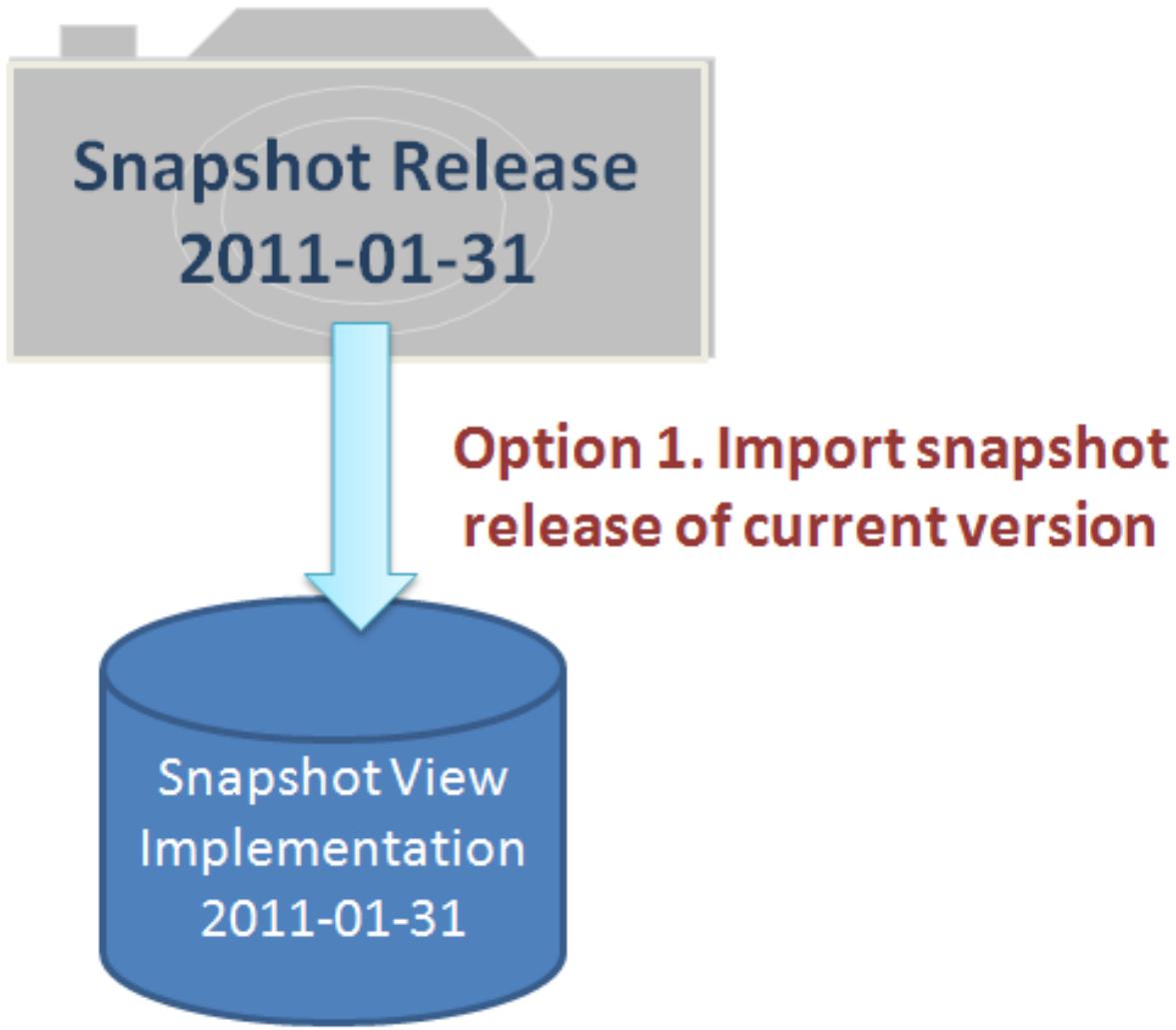


Figure 79: Initial import to create a snapshot view

2. Create a snapshot from a set of *full release* files for a version of the *International Release*:
 - When creating a snapshot from a *full release*, only those rows that represent the most recent version of each *component* are imported.
 - Where two or more rows have the same id, only the row with the most recent *effectiveTime* is imported into the snapshot.
- ⚠ Caution:** Only take account of the id and *effectiveTime* fields when determining which rows to import into a snapshot. A common mistake is to look for the most recent active row. This results in serious errors. The active field should only be considered after importing the data and then provides information on whether that *component* is or is not active as part of the snapshot.

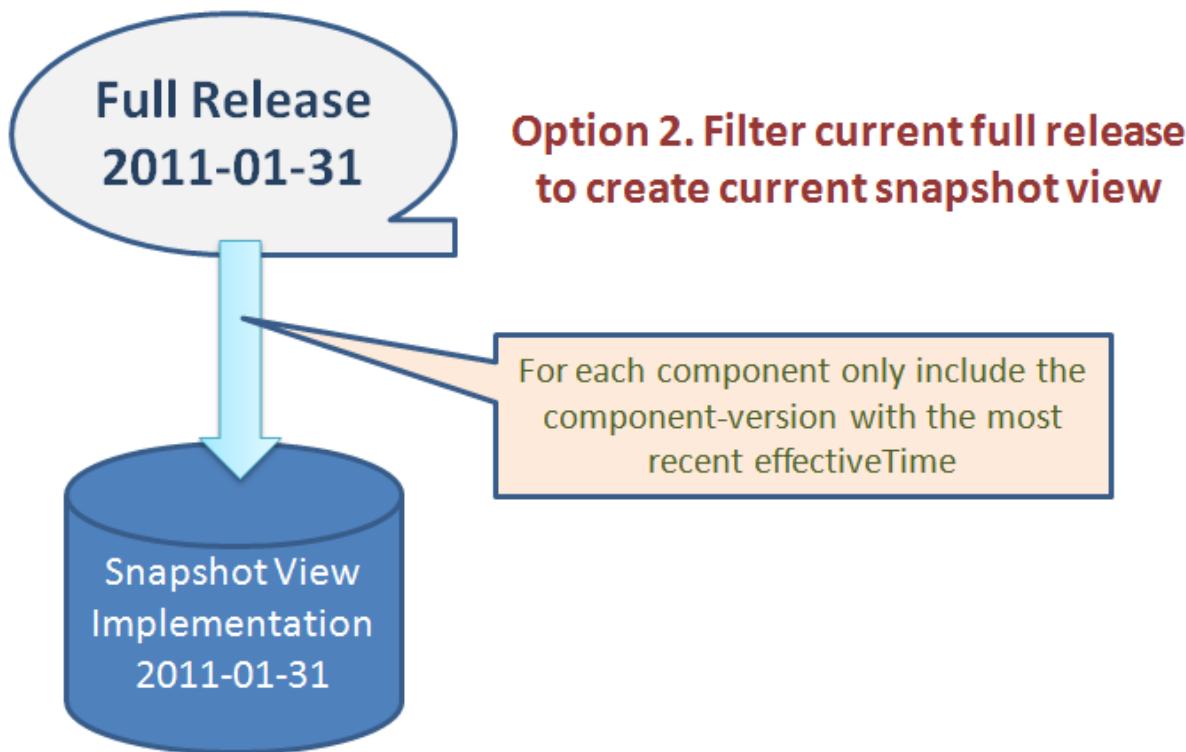


Figure 80: Initial import to create a snapshot view from a full release

👉 **Note:** Option 2 can also be used to create an earlier *snapshot view*. To do this only import rows that represent the most recent version of each *component* with an *effectiveTime* that is no later than the time of the required snapshot.

👉 **Tip:** The files that form part of a particular release can be identified by pattern matching based on the IHTSDO filenaming conventions (see [Identifying release files using regular expressions](#)).

7.2.1.3.2 Updating a Snapshot view



A *snapshot view* can be updated by one of the following approaches:

1. Use a set of *delta release* files to update the a *snapshot view* of the previous version:
 - The overall process can be described as follows:
 - Append the *delta release* to the previous snapshot;
 - Filter to remove rows that have the same id so that only the row with the most recent *effectiveTime* remains.
 - An efficient way to achieve this end result is to take account of the fact that the most recent version of any given *component* will be in the new *delta release* rather than in the previous version of the *snapshot view*.

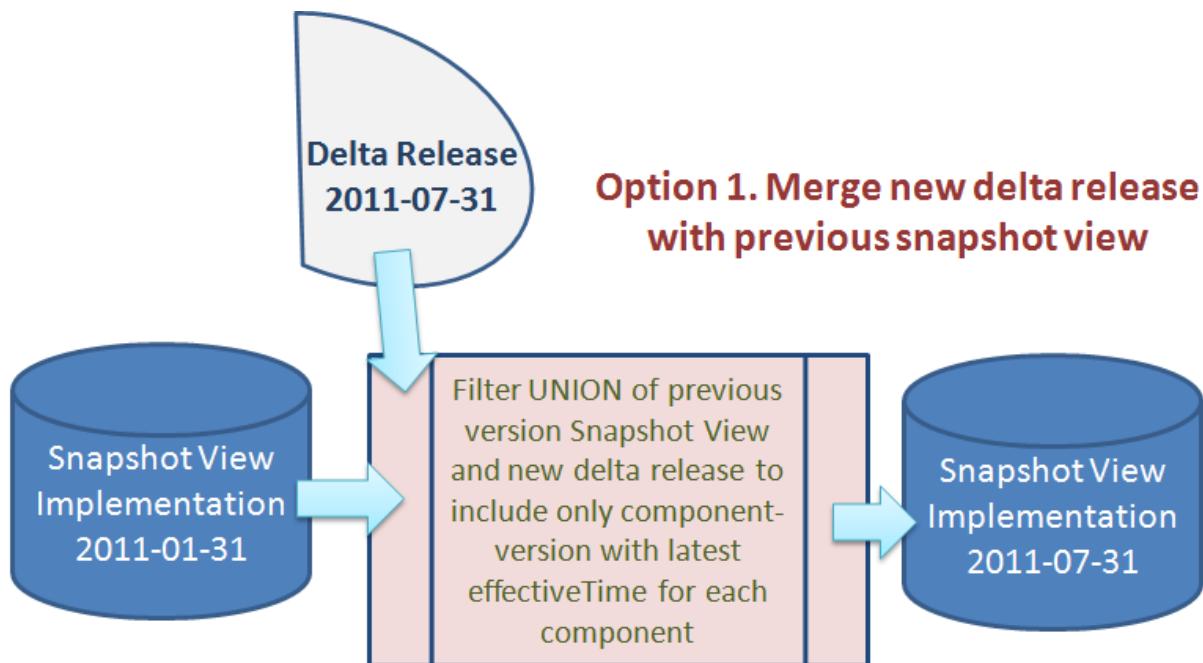


Figure 81: Updating a snapshot view using a delta release

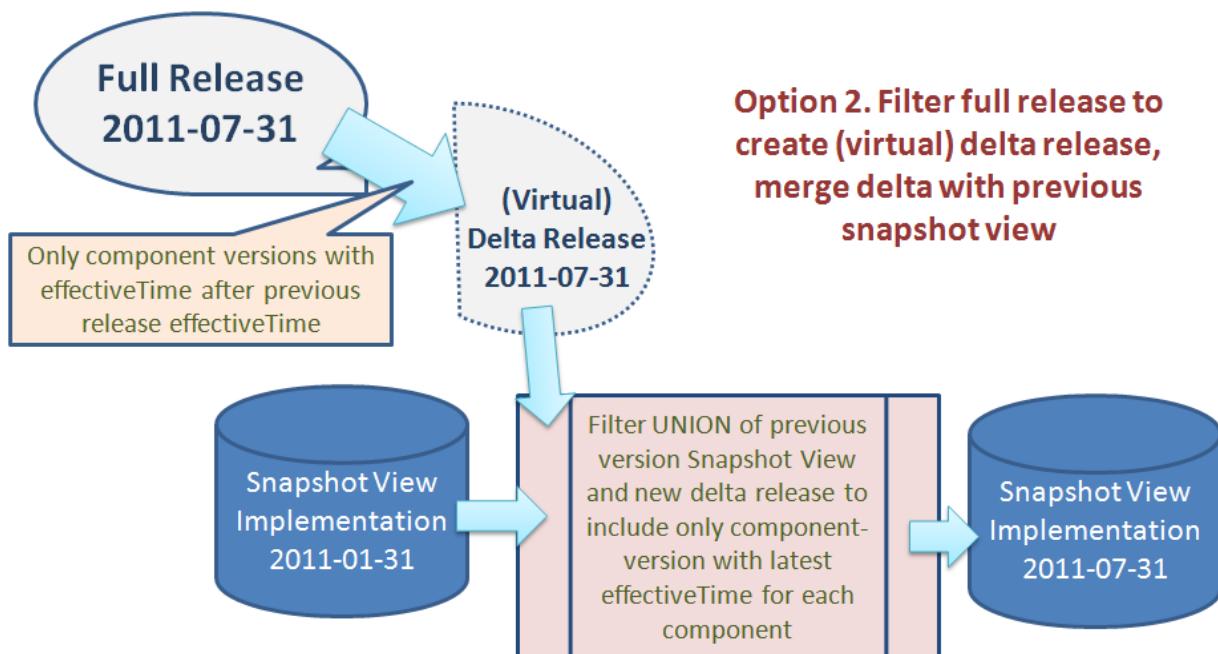


Figure 82: Updating a snapshot view using a full release

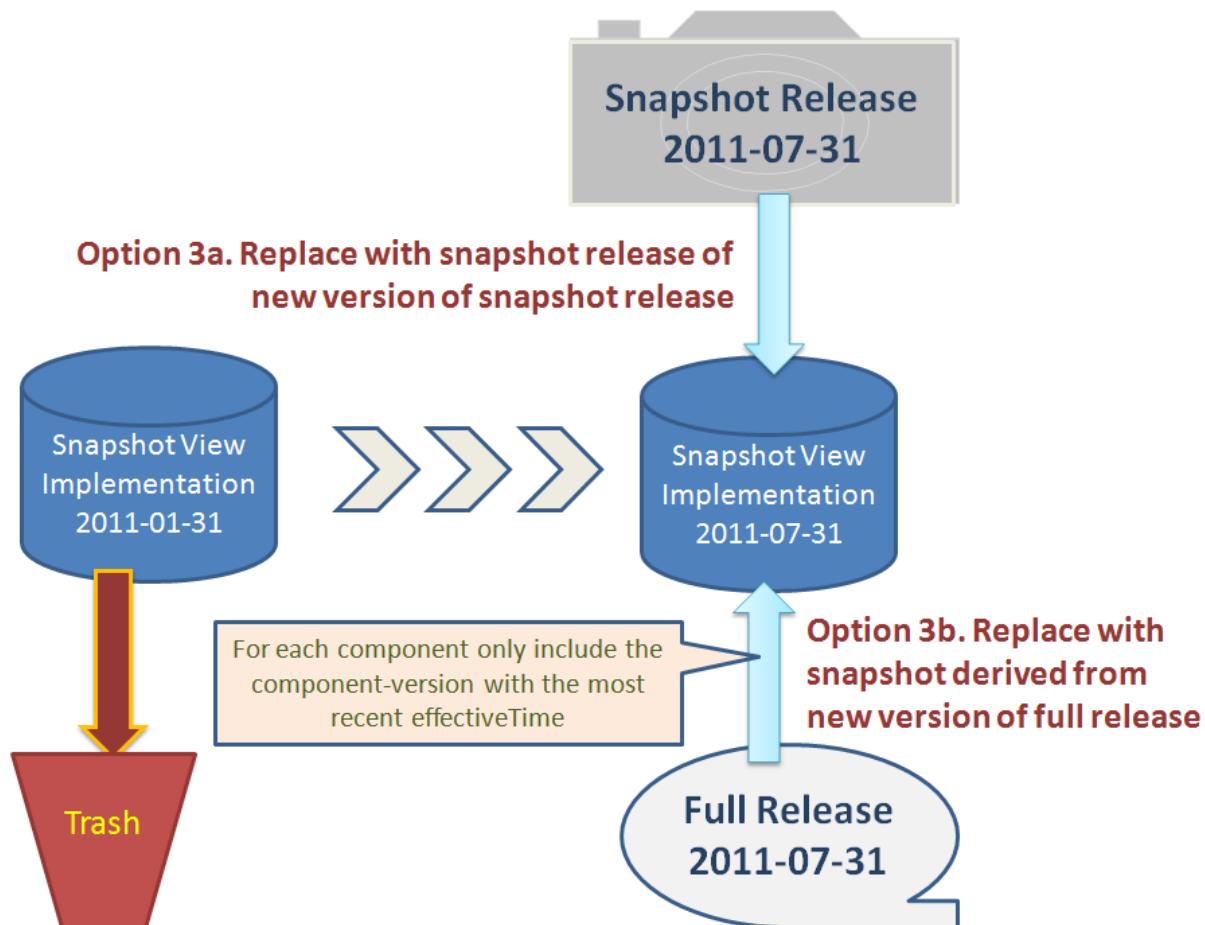


Figure 83: Updating a snapshot view by replacement

👉 **Tip:** The files that form part of a particular release can be identified by pattern matching based on the IHTSDO filenaming conventions (see [Identifying release files using regular expressions](#)).

7.2.1.3.3 Importing and updating Extensions for a Snapshot view



To provide access to the *snapshot view* of one or more *Extensions*, a *terminology server* must initially import or create the current snapshot for each required *Extension*. This can be done either using *snapshot release* files or *full release* files as described for [importing a Snapshot view](#). Thereafter, the *snapshot view* of each *Extension* can be maintained by using any of the techniques described for [updating a Snapshot view](#).

When a *snapshot view* of *Extension* data is initially imported or subsequently updated care needs to be taken to ensure the relevant versions of the *International Release* and any other *Extensions* on which it depends have also been imported or updated. Failure to do this will lead to errors as a result of references from *Extension components* to missing or out of date *components* in the *International Release* or in another *Extension*.

For all normal uses the *snapshot view* of an *Extension* version must be combined with the *snapshot view* of the versions of the *International Release* on which it was based. Similarly, if an *Extension* is dependent on another *Extension*, the *snapshot* of the *Extension* on which it depends must be for the version on which the dependent *Extension* version was based.

The table below summarizes the compatibility between the *snapshot views* of given versions of an *Extension* and the *International Release*. It also identifies some limited cases in which a *snapshot view* may be used when not directly compatible with the relevant *International Release* snapshot. If one *Extension* depends on another *Extension*, the same considerations apply to compatibility between the versions of those *Extensions*.

Table 179: Compatibility between snapshot views of versions of an Extension and the International Release

Relationship between the version of the <i>Extension</i> and the <i>International Edition</i>	Notes on compatibility and usability
Installed <i>International Release</i> is older than the version on which the <i>Extension</i> was based	<p>Incompatible.</p> <p>The <i>Extension</i> may include <i>Relationships</i> to concepts that do not exist in this version of the <i>International Release</i>. This will lead to errors that cannot be reconciled while viewing the <i>Extension</i> content. Excluding <i>components</i> that conflict in this way results in other errors the previous state of the <i>Extension</i> content is not available in a <i>snapshot view</i>.</p> <p>A system with this mix of snapshot versions cannot be safely used.</p>
Installed <i>International Release</i> is same version as the one on which the <i>Extension</i> was based	<p>Full compatible.</p> <p>This is the version the <i>Extension</i> was created for so it should behave as intended.</p>
Installed <i>International Release</i> is newer than the version on which the <i>Extension</i> was based.	<p>Partially compatible - subject to appropriate configuration and usage.</p> <p>The <i>International Edition</i> for this version may include:</p> <ul style="list-style-type: none"> • Additional <i>components</i>. These will not cause errors because the <i>International Release</i> does not reference the <i>Extension</i> and the <i>Extension</i> content cannot reference <i>components</i> that did not exist when in the version it was based on. • Changes to the state of some <i>components</i>. These changes may affect the interpretation of some parts of the <i>Extension</i>. <p>In a <i>snapshot view</i> the <i>International Release</i> cannot be rolled back to its previous state and, as a result, cannot be aligned with the version on which the <i>Extension</i> was based. Therefore, the potential for safe use of combinations of this is limited to the following:</p> <ol style="list-style-type: none"> 1. Configured to exclude the <i>Extension</i>. In this case the most recent snapshot of the <i>International Release</i> can be viewed. The incompatible <i>Extension</i> is ignored. 2. Configured to use those parts of the <i>Extension</i> that support translation of <i>International Release</i> content. In this case, the <i>Extension</i> will enable translated rendering of pre-existing translated content. This would leave new and untranslated <i>concepts</i> to be rendered in English (or another available language).

 **Note:** A *full view* implementation can be configured to be more tolerant to different versions of installed *Extensions* and *International Releases*. In effect the *full view* allows virtual snapshots of the state of each *Extension* to be used to deliver a compatible set of *component-versions*.

7.2.1.4 Maintaining a Multi-snapshot view



If more than one *snapshot view* is required, the most effective approach is to implement a *full view* that enables a *dynamic snapshot* to be provided for any chosen time. The alternative approach is to

create several separate *snapshot views* and to allow users to choose and where necessary switch between these static snapshots.

Each of these views in a multi-snapshot view is separately created and maintained in the same way as a single *snapshot view*. The required view for a particular purpose is selected from those available in the server. Where necessary more than one view may be selected to identify changes between versions.

In the long-term this approach requires more maintenance effort and more storage space than a *full view* and is far less flexible. It assumes a small set of discrete views such as those that arise from a relatively infrequent releases of *SNOMED CT* content. A more gradual evolution of content may occur in future as a result of the additions to *Extensions* and the ability to distribute *delta releases*. The multiple-snapshot approach may still meet the limited requirements of an organization needing access to two or three specified *snapshot views* (e.g. for current, previous and perhaps one other defined reference date). This approach may be useful as an interim measure in an environment that is unable to provide adequate performance for *dynamic snapshot views*.

7.2.2 Choosing the release files to import



The *International Release* files to be imported should all be selected from the set of files representing a single *Release Type* for a chosen version of the *SNOMED CT International Release*.

Within the chosen file set the files identified in *Table 180* must be imported. The files listed in *Table 181* should also be imported as these provide important information about *inactive concepts* and metadata about *Description* types. The decision on whether to import the files listed in *Table 182* depends on whether the additional features identified in that table are required for the planned implementation. Finally the supplementary files listed in *Table 183* may be used to assist implementation but are not essential as the data they contain can be generated from the other files and/or replaced by alternative approaches to provide similar functionality.

Table 180: Mandatory import files

File type	Content	Notes
sct2_concept_[rt]_INT...	concepts	The primary components of <i>SNOMED CT</i> . Essential for all implementations.
sct2_description_[rt]_INT...	Descriptions	
sct2_relationship_[rt]_INT...	Relationships	
sct2_cRefset_Language [rt]-[lang]_INT...	Language Refset(s)	At least one Language Refset must be imported. The English Language refset should be imported unless another Language refset covering the full content is available and imported.

Table 181: Highly recommended import files

File type	Content	Notes
der2_cRefset_AttributeValue [rt]_...	concept Inactivation refset	Provides information about the status of <i>inactive concepts</i> .
	Description Inactivation Refset	Provides information about the status of <i>inactive descriptions</i> .

File type	Content	Notes
der2_cRefset_DescriptionType [rt]_INT...		
der2_cRefset_AssociationReference [rt]_INT...		

Table 182: Optional import files

File type	Content	Notes
der2_sRefset_SimpleMap [rt]_INT...	Maps from <i>NHS Clinical Terms Version 3</i> codes, other <i>Read Codes</i> to <i>SNOMED CT</i>	Only required if the server needs to be able to lookup <i>SNOMED CT concepts</i> based on a <i>CTV3 Identifier</i> or <i>Read Code</i> .
	Maps from legacy <i>SNOMED CT 3</i> codes to <i>SNOMED CT</i>	Only required if the server needs to be able to lookup <i>SNOMED CT concepts</i> based on a legacy <i>SNOMED CT 3</i> code.
WordEquivalents		
StatedRelationships		

Table 183: Supplementary import files

File type	Content	Notes
DescWordKey		
DescDualKey		
ExcludedWords		
TransitiveClosure		Generated from the <i>Relationships</i> . Needs to be regenerated or updated if <i>Extensions</i> are imported.

If *Extensions* are required to support an implementation, the *release files* to be imported should be selected from the set of files for a single *Release Type* for a chosen version of that *Extension*. It is important to ensure that the *International Release* version(s) on which all the imported *Extensions* are based has also been imported. The files that need to be imported from a chosen *Extension* may vary depending on the scope of the *Extension*.

 **Note:** Advice should be sought from the *Extension* provider on the essential and recommended requirements of files to be imported and supported.

7.2.3 Choosing extension files to import



The process of importing an *Extension* is similar to importing the main distribution files. However, some additional functionality is required to ensure appropriate installation, maintenance and use of *Extensions*.

Applications should:

- Allow the users or user communities to specify the *Extensions* to be recognized by their systems. Before recognizing any *Extension*, users should check that:
 - The *Extension* has been supplied by the *IHTSDO* or another organization authorized by the *IHTSDO* to provide such *Extensions*.
 - You are satisfied with the quality control procedures of the providing organization:
 - Authorization of an organization to produce *Extensions* does not imply any seal of approval related to the quality of *Extensions* provided by those organizations;
 - Installation of *Extensions* is done entirely at the risk of the user subject to their license agreement with the provider of the *Extension* and/or the application developer.

7.2.4 Identifying release files using regular expressions



The files that form part of each release follow *IHTSDO* file naming conventions. These conventions allow the files that form part of a particular *Release Type*, version or extension to be identified by pattern matching. The following tables include examples of [standard regular expressions](#) that selectively match particular sets of *release files*.

Table 184: General patterns for Release Types

<i>Release</i>	<i>Type</i>	<i>Regular Expression</i>
International	full	$\wedge x? (sct der res) 2_[^_]+\[_\]^* Full([-a-z-\{2,6\})? _{INT_2}[0-9]\{7\}.txt\$$
International	delta	$\wedge x? (sct der res) 2_[^_]+\[_\]^* Delta(-[a-z-\{2,6\})? _{INT_2}[0-9]\{7\}.txt\$$
International	snapshot	$\wedge x? (sct der res) 2_[^_]+\[_\]^* Snapshot(-[a-z-\{2,6\})? _{INT_2}[0-9]\{7\}.txt\$$
Any Extension	full	$\wedge x? (sct der res) 2_[^_]+\[_\]^* Full([-a-z-\{2,6\})? _{([A-Z]\{2\})? [0-9]\{7\}_2}[0-9]\{7\}.txt\$$
Any Extension	delta	$\wedge x? (sct der res) 2_[^_]+\[_\]^* Delta(-[a-z-\{2,6\})? _{([A-Z]\{2\})? [0-9]\{7\}_2}[0-9]\{7\}.txt\$$
Any Extension	snapshot	$\wedge x? (sct der res) 2_[^_]+\[_\]^* Snapshot(-[a-z-\{2,6\})? _{([A-Z]\{2\})? [0-9]\{7\}_2}[0-9]\{7\}.txt\$$

Table 185: Example patterns for specific versions and extensions

<i>Release</i>	<i>Regular Expression</i>
International Type: full Version: 2010-07-31	<code>\^x?(sct der res)2_\[^_]+_\[^_]*Full(-[a-z-]\{2,6\})?_INT_20100731.txt\$</code>
International Type: full Version: 2011-01-31	<code>\^x?(sct der res)2_\[^_]+_\[^_]*Full(-[a-z-]\{2,6\})?_INT_20110131.txt\$</code>
<i>Member Extension</i> Type: full Country: GB Namespace: 1000001 Version: 2011-04-01	<code>\^x?(sct der res)2_\[^_]+_\[^_]*Full(-[a-z-]\{2,6\})?_GB10000001_20110401.txt\$</code>
<i>Affiliate Extension</i> Type: full Namespace: 1000003 Version: 2011-07-31	<code>\^x?(sct der res)2_\[^_]+_\[^_]*Full(-[a-z-]\{2,6\})?_10000003_20110731.txt\$</code>

7.2.5 Checking during the import process



The import process should check the imported data to confirm that:

- The distribution files imported are all part of the same release.
- The set of files imported is complete and includes all mandatory components.
- In the case of a *delta release*, the data previously imported is the version immediately prior to the *delta release* being imported.
- 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*.
- All component *Identifiers* have:
 - A *partition identifier* appropriate to the type of component;
 - A valid *check-digit*.
- All fields meet data type, size and value constraints specified for the relevant tables.

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

7.2.5.1 Additional checks when importing Extensions



The process of importing an *Extension* is similar to importing the main distribution files. However, some additional functionality is required to ensure appropriate installation, maintenance and use of *Extensions*. Applications should:

Check each *Extension* prior to installation to ensure that:

- It is one of the *Extensions* recognized by the user.
 - It is supported by or based on the currently installed *International Release* version.
 - 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).
 - Any dependencies of the *Extension* have been met. These dependencies may include:
 - Installation of a particular *SNOMED CT release*;
 - Prior installation of other *Extensions*.
- Note:** Dependencies are represented using the *moduleId* and the *Module Dependency Reference Set*.
- The installation procedure has pre-checked all *components* in the *Extension* to ensure that:
 - All Component *Identifiers*:
 - Are unique;
 - 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.

Caution: If any *components* fail any of these tests the entire *Extension* must 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 the *Extension* and the *International release*.

- Reject, highlight or apply other agreed business rules to information received by the system that contains *SCTIDs* for *components* from namespaces that are not in the list, or recognized *Extensions*.

7.2.6 Pre-processing of distribution files by terminology server suppliers



The import process may be time-consuming due to the need to build indices or other data structures. It may also require substantial spare storage capacity for temporary files. Therefore a *terminology server* provider may choose to pre-import the distribution files and provide them to users in pre-prepared form. However, an import facility should also be available in a suitably secured form to end-user organizations, to enable installation and maintenance of *Extensions*.

7.3 Implementing Dynamic Snapshot Views



A key feature of *SNOMED CT Release Format 2* is that it allows a single database table to represent the *full view* of a *SNOMED CT component*. This view includes all versions of the *component* from its first release up to its state in the latest release. This offers several significant benefits which are described elsewhere in the guide.

Most frequently used *SNOMED CT* functions need to provide access to a 'snapshot' view of the content of *SNOMED CT* at a point in time.

- Everyday use of *SNOMED CT* for data entry and retrieval will generally require a current 'snapshot' view.

 **Example:** To see the active content of *SNOMED CT* including all the most up to date *components* and excluding any *components* that have been marked as inactive.

- There are some situations in which a retrospective 'snapshot' view of the data at a selected point in the past is required.

 **Example:**

To see the definition of a *concept* as it was when a record entry was created.

To see the version of the *International Release* on which the latest available version of an *Extension* was based.

7.3.1 Generating Dynamic Snapshot Views



The general method for creating a snapshot 'view' for a specified SnapshotTime is as follows:

1. Exclude all *Component* versions with an *effectiveTime* greater than the SnapshotTime.

 **Note:** In theory the most recent *snapshot* view step could be omitted. However, a release will often be distributed before its *effectiveTime*. Therefore, this approach is not recommended as a general approach in a live system.

2. From each set of *Component* versions with the same id select the *Component* version with the highest (most recent) *effectiveTime*.

The most flexible approach is to apply this method dynamically so that a different snapshot time can be configured as needed to meet new requirements. The following example code illustrates an implementable approach to this.

```
SELECT `c`.* FROM `sct2_concept` AS `c`
WHERE `c`.`effectiveTime` = (SELECT MAX(`c2`.`effectiveTime`)
                           FROM `sct2_concept` `c2`
```

```
WHERE `c2`.`id` = `c`.`id`
AND `c2`.`effectiveTime` <= `snapshotTime`()
```

Figure 84: General form of SQL to create a snapshot view

In this sample code `snapshotTime()` is a function that returns the time to be applied to this snapshot. For the most recent *snapshot view* this can be omitted as shown below:

```
SELECT `c`.*
FROM `sct2_concept` AS `c`
WHERE `c`.`effectiveTime` = (SELECT MAX(`c2`.`effectiveTime`)
                           FROM `sct2_concept` `c2`
                           WHERE `c2`.`id` = `c`.`id`)
```

Figure 85: SQL to create the latest snapshot view

Similar views can be created for each of the *Component* tables by simply replacing the table name in both the outer and nested queries.

7.3.2 Optimizing Dynamic Snapshots Views



Some databases may be able to generate dynamic *snapshot views* sufficiently rapidly to enable real time use. However, in other cases, even if the nested queries used in the general *snapshot views* work quickly on their own, more complex queries involving joins between different *component* tables may lead to performance degradation. There are several approaches that can be taken to optimizing performance and two of these are in the following subsections.

7.3.2.1 Optimizing using a Snapshot View Flag



The first optimization approach is provide a simple way to optimize the current snapshot and can be extended to cover a limited number of additional *snapshot views*. A column is added to each *component* table to hold a *boolean* value that indicates whether or not a particular row is part of the current snapshot. In the following *Description* and example this added column is called `inSnapshot` and is referred to as a "snapshot view flag".

After importing or updating *SNOMED CT* content the *snapshot view flag* is updated using the results of a *snapshot view query* such as one illustrated in [Figure 86](#). The example uses an intermediate temporary table. In some relational database environments nested queries could be used to reduce the number of steps in the script. However, the longer form is used here as some environments do not work (or are unpredictable) when updating a table that is also referenced by a nested select query.

```
/* Clear the inSnapshot flag */
UPDATE `sct2_concept` SET `inSnapshot`=False;

/* Create temporary table to hold latest id+effectiveTime */
DROP TEMPORARY TABLE IF EXISTS `tmp_ids`;

CREATE TEMPORARY TABLE `tmp_ids` (`id` BIGINT,`effectiveTime` DATETIME, PRIMARY KEY (`id`));

/* replace the line above with the line below for Refsets as the Id is a UUID rather than SCTID */
/* CREATE TEMPORARY TABLE `tmp_ids` (`id` BINARY(16),`effectiveTime` DATETIME, PRIMARY KEY (`id`)); */

/* Populate the temporary table with id+effectiveTime for the latest view*/
INSERT INTO `tmp_ids` SELECT `id`, `effectiveTime` FROM `sct2_concept` AS `c`
WHERE `c`.`effectiveTime` = (SELECT MAX(`c2`.`effectiveTime`)
                           FROM `sct2_concept` `c2`
                           WHERE `c2`.`id` = `c`.`id`);

/* Use the temporary table to update the inSnapshot flag for relevant rows */
UPDATE `sct2_concept` AS `c`, `tmp_ids` AS `t`
SET
`inSnapshot` = True
```

```

WHERE `c`.`id` = `t`.`id` AND `c`.`effectiveTime` = `t`.`effectiveTime`;

/* Clean up by removing the temporary table */
DROP TEMPORARY TABLE `tmp_ids`;

```

Figure 86: Setting the latest snapshot view flag

The following query illustrates the simple query that can be used to return the current *snapshot view* using the *snapshot view* flag.

```
SELECT `c`.* FROM `sct2_concept` AS `c` WHERE `c`.inSnapshot` = True;
```

Figure 87: Using a snapshot view flag to select components in a snapshot view

The same approach can be applied to each of the *components* by replacing `sct2_concept` with the relevant table name.

Additional *snapshot view* flags can be added, set and used in a similar way for a few other snapshot times that need to be optimized.

```

/* Clear the inSnapshotPrev flag */
UPDATE `sct2_concept` SET `inSnapshotPrev` = False;

/* Create temporary table to hold latest id+effectiveTime */
DROP TEMPORARY TABLE IF EXISTS `tmp_ids`;

CREATE TEMPORARY TABLE `tmp_ids`(`id` BIGINT, `effectiveTime` DATETIME, PRIMARY KEY (`id`));

/* replace the line above with the line below for Refsets as the Id is a UUID rather than SCTID */
/* CREATE TEMPORARY TABLE `tmp_ids`(`id` BINARY(16), `effectiveTime` DATETIME, PRIMARY KEY (`id`));

/* Populate the temporary table with id+effectiveTime for the specified view date time */
INSERT INTO `tmp_ids` SELECT `id`, `effectiveTime` FROM `sct2_concept` AS `c`
    WHERE `c`.`effectiveTime` = (SELECT MAX(`c2`.`effectiveTime`)
        FROM `sct2_concept` `c2`
        WHERE `c2`.`id` = `c`.`id` AND `c2`.`effectiveTime` <= CAST('2010-01-31', DATETIME));

/* Use the temporary table to update the inSnapshotPrev flag for relevant rows */
UPDATE `sct2_concept` AS `c`, `tmp_ids` AS `t`
SET
    `inSnapshotPrev` = True
    WHERE `c`.`id` = `t`.`id` AND `c`.`effectiveTime` = `t`.`effectiveTime`;

/* Clean up by removing the temporary table */
DROP TEMPORARY TABLE `tmp_ids`;

```

Figure 88: Setting the snapshot view flag for a specified date

This approach provides a simple approach to optimization of a limited number of views. However, it is constrained by the need to allocate a column for each time for which an optimized *snapshot view* is required.

7.3.2.2 Optimizing using a Superseded Time



This approach to optimization of *dynamic snapshot views* uses a single additional column in each *component* table to denote the time at which a row was superseded by a new version of the same *component*. This is more flexible but may not deliver the same performance improvement as the *snapshot view* flag approach.

After importing or updating *SNOMED CT* content the superseded time values are checked and updated where relevant using a query such as one illustrated in [Figure 89](#). In this example, a fixed distant future date (31-12-9999) is used for *Components* which have not been superseded. The alternative would be a null date

but the fixed distant date avoids the need to look for null as an exception at runtime. It also allows additional optimization of the current view - particularly if the supersededTime is indexed.

```

/* Create temporary table to hold latest id+effectiveTime */
DROP TEMPORARY TABLE IF EXISTS `tmp_supersede`;

CREATE TEMPORARY TABLE `tmp_supersede` (`id` BIGINT,`effectiveTime` DATETIME,`supersededTime` DATETIME, PRIMARY KEY (`id`, `effectiveTime`));

/* replace the line above with the line below for Refsets as the Id is a UUID rather than SCTID */
/* CREATE TEMPORARY TABLE `tmp_supersede` (`id` BINARY(16),`effectiveTime` DATETIME,`supersededTime` DATETIME, PRIMARY KEY (`id`, `effectiveTime`)); */

/* Populate the temporary table with id+effectiveTime+supersededTime */
INSERT INTO `tmp_supersede`
    SELECT `c`.`id`, `c`.`effectiveTime`, (SELECT IFNULL(MIN(`c2`.`effectiveTime`),CAST('9999-12-31' AS DATETIME))
    FROM `sct2_concept` AS `c2` WHERE `c`.`id` = `c2`.`id` AND `c`.`effectiveTime` < `c2`.`effectiveTime`) AS supersededTime
    FROM `sct2_concept` AS `c`;

/* Use the temporary table to update the supersededTime flag for relevant rows */
UPDATE `sct2_concept` AS `c` JOIN `tmp_supersede` AS `t`
    ON `t`.`id` = `c`.`id` AND `t`.`effectiveTime` = `c`.`effectiveTime`
    SET `c`.`supersededTime` = `t`.`supersededTime`;

/* Clean up by removing the temporary table */
DROP TEMPORARY TABLE `tmp_supersede`;

```

Figure 89: Populating or updating the superseded time after importing content

Figure 90 illustrates the general query for returning the *snapshot view* for a specified time. To be included in the view the *effectiveTime* must be the same as or before the snapshot time and the *supersededTime* must be after the snapshot time.

```
SELECT `c`.* FROM `sct2_concept` AS `c` WHERE `c`.`effectiveTime` <= `snapshotTime`() AND
`c`.`supersededTime` > `snapshotTime`();
```

Figure 90: Using the superseded time to select components in the a specified snapshot view

Figure 91 illustrates the simpler query that can be used to return the current *snapshot view* using the superseded time value. If the supersededTime is included in the relevant composite indexes this may further improve the optimization for this commonly required view.

```
SELECT `c`.* FROM `sct2_concept` AS `c` WHERE `c`.`supersededTime` = CAST('9999-12-31' AS DATETIME);
```

Figure 91: Using the superseded time to select components in the current snapshot view

The same approach can be applied to each of the *Components* by replacing `sct2_concept` with the relevant table name.

7.4 Working with metadata



SNOMED CT RF2 files represent some key information about core release *components* by reference to other *SNOMED CT components*. Two types of metadata (*Concept Enumerations* and *Reference Sets*) are described in the following sections:

- *Concept Enumerations* provide sets of values for enumerated fields in *SNOMED CT components*.

- Reference Sets support a wide range of functions and the following section subdivide these functions according their relative importance:
 - Essential Reference Sets;
 - Optional Reference Sets;
 - Reference Sets supporting advanced functionality.
 - <xml></xml>

7.4.1 Concept enumerations



SNOMED CT core components have some fields that have values represented by *concepts* in specific parts of the *SNOMED CT* hierarchy. These are referred to as *concept* enumerations.

The range of permitted values for each of the *concept* enumerations is the set of *subtypes* of a specified *concept* which is itself a *subtype* of 900000000000442005 | Core metadata concept (core metadata concept) |. The current set of *concept* enumeration types is shown in *Table 186*. The values of each of these and the ways they should be used in implemented systems are described in the following subsections.

Table 186: Core metadata concept (core metadata concept) (900000000000442005)

Id	Term	Comment
900000000000443000	Module (core metadata concept)	Each <i>subtype</i> of this <i>concept</i> represents a development module. These <i>concepts</i> provide values to the <i>moduleId</i> field that is present in all <i>SNOMED CT component</i> file. The value indicates the module within which a <i>component</i> was created and is being maintained.
900000000000444006	Definition status (core metadata concept)	Each <i>subtype</i> of this <i>concept</i> represents a value that can be applied to the <i>concept.definitionStatusId</i> field. This is used to indicate whether the current set of defining <i>Relationships</i> applied to a <i>concept</i> are sufficient to fully-define it relative to its supertypes.
900000000000446008	Description type (core metadata concept)	Each <i>subtype</i> of this <i>concept</i> represents a value that can be applied to the <i>Description.descriptionTypeId</i> field. This is used to indicate whether the <i>Description</i> represents a <i>Fully Specified Name</i> , a synonymous term, a definition or some other symbolic or textual representation of the associated <i>concept</i> .

Id	Term	Comment
9000000000000447004	Case significance (core metadata concept)	Each <i>subtype</i> of this <i>concept</i> represents a value that can be applied to the <i>Description.caseSignificanceId</i> field. This is used to indicate whether the text of the term can be modified to by switching characters from upper to lower case (or vice-versa).
9000000000000449001	Characteristic type (core metadata concept)	Each <i>subtype</i> of this <i>concept</i> represents a value that can be applied to the <i>Relationship.characteristicTypeId</i> field. This is used to indicate whether a <i>Relationship</i> forms part of the definition of the source <i>concept</i> .
9000000000000450001	Modifier (core metadata concept)	Each <i>subtype</i> of this <i>concept</i> represents a value that can be applied to the <i>Relationship.modifierId</i> field. This is used to indicate the type of <i>Description Logic</i> (DL) restriction (some, all, etc.) that applies to the <i>Relationship</i> .
9000000000000453004	Identifier scheme (core metadata concept)	Each <i>subtype</i> of this <i>concept</i> represents a value that can be applied to the <i>Identifier.identifierSchemeId</i> field. This is used to indicate the scheme to which the <i>Identifier</i> value belongs.

 **Note:** Many of the *concept* enumerations include values that significantly impact the meaning or use of a *component*. Therefore, implementers may find it necessary to partially hard-code the way their systems process particular values. In these cases, the *concept* referenced by the value is only of value when there is a requirement to display a human readable rendering of the value. The main exceptions to this are 900000000000443000 | Module (core metadata concept) | and 900000000000453004 | Identifier scheme (core metadata concept) | both of which represent extensible sets of values as new modules or alternative *Identifier* schemes may be added in local *Extensions*.

7.4.1.1 Concept enumerations for moduleId



This *concept* enumeration applies to the *moduleId* field which is present in all released *SNOMED CT components* (RF2). The value applied to a particular *component* indicates the development module within which that *component* was created and is being maintained.

Each of the values in [Table 187](#) represents a development module. The range of permitted list of values is extensible by addition of branches to the hierarchy shown in [Figure 92](#) modules managed by other organizations (i.e. in an extensions namespace) and to add specific module *Identifiers* within each branch.

Table 187: International Health Terminology Standards Development Organisation maintained module (core metadata concept) (900000000000445007)

Id	Term	Comment
90000000000000012004	SNOMED CT model component module (core metadata concept)	
900000000000000207008	SNOMED CT core module (core metadata concept)	

- 900000000000443000 | Module (core metadata concept) |
 - 900000000000445007 | International Health Terminology Standards Development Organisation maintained module (core metadata concept) |
 - 90000000000000012004 | SNOMED CT model component module (core metadata concept) |
 - 900000000000000207008 | SNOMED CT core module (core metadata concept) |

Figure 92: Hierarchy of SNOMED CT moduleId values

7.4.1.2 Concept enumerations for definitionStatusId



This *concept* enumeration represents a value that can be applied to the *concept.definitionStatusId* field. This is used to indicate whether the current set of defining *Relationships* applied to a *concept* are sufficient to fully-define it relative to its supertypes.

Table 188 shows the current set of values for this *concept* enumeration.

Table 188: Definition status (core metadata concept) (900000000000444006)

Id	Term	Comment
90000000000000073002	Sufficiently defined concept definition status (core metadata concept)	<p>The set of defining <i>Relationships</i> applied to the <i>concept</i> are asserted to fully define the <i>concept</i>.</p> <p>Any <i>concept</i> or <i>expression</i> for which all these defining <i>Relationships</i> are true is either equivalent to or subsumed by this <i>concept</i>.</p> <p>Any <i>concept</i> or <i>expression</i> for which any of these defining <i>Relationships</i> is not true is neither equivalent to nor subsumed by this <i>concept</i>.</p>

Id	Term	Comment
90000000000000074008	Necessary but not sufficient concept definition status (core metadata concept) 	<p>The set of defining <i>Relationships</i> applied to the <i>concept</i> are asserted to be incompletely define the <i>concept</i>. The <i>concept</i> is currently considered to be <i>primitive</i>.</p> <p>A <i>concept</i> or <i>expression</i> for which all these defining <i>Relationships</i> are true may be equivalent to or subsumed by this <i>concept</i>. However, it is not possible to compute this from the definition - because the missing element in the definition may or may not apply to the other <i>concept</i> or <i>expression</i>.</p> <p>Any <i>concept</i> or <i>expression</i> for which any of these defining <i>Relationships</i> is not true is neither equivalent to nor subsumed by this <i>concept</i>.</p>

7.4.1.3 Concept enumerations for descriptionTypeIId



This *concept* enumeration represents a value that can be applied to the *Description*.
descriptionTypeIId field. This is used to indicate whether the *Description* represents a *Fully Specified Name*, a synonymous term, a definition or some other symbolic or textual representation of the associated *concept*.

Table 189 shows the current set of values for this *concept* enumeration.

Table 189: Description type (core metadata concept) (900000000000446008)

Id	Term	Comment
9000000000000003001	Fully specified name (core metadata concept) 	The <i>Description.term</i> represents the <i>Fully Specified Name</i> of the associated <i>concept</i> in the language indicated by the <i>Description.languageCode</i> .
90000000000000013009	Synonym (core metadata concept) 	<p>The <i>Description.term</i> represents a term that is used to represent the associated <i>concept</i> in the language indicated by the <i>Description.languageCode</i>.</p> <p> Note: The <i>preferred term</i> used in a given language or <i>dialect</i> is marked as a <i>synonym</i>. Preference and acceptability of a particular synonymous term is indicated by a <i>Language refset</i>.</p>
9000000000000550004	Definition (core metadata concept) 	The <i>Description.term</i> represents a textual definition of the associated <i>concept</i> in the language indicated by <i>Description.languageCode</i> .

7.4.1.4 Concept enumerations for caseSignificanceId



This *concept* enumeration represents a value that can be applied to the *Description.caseSignificanceId* field. This is used to indicate whether the text of the term can be modified to by switching characters from upper to lower case (or vice-versa).

Table 190 shows the current set of values for this *concept* enumeration.

Table 190: Case significance (core metadata concept) (900000000000447004)

Id	Term	Comment
90000000000000017005	Entire term case sensitive (core metadata concept)	The text of the <i>Description.term</i> must be presented in the case in which it is specified.
90000000000000020002	Only initial character case insensitive (core metadata concept)	The initial character of the <i>Description.term</i> is case insensitive and can be changed from upper to lower case (or vice-versa) if appropriate to the context in which it is used.
900000000000448009	Entire term case insensitive (core metadata concept)	The entire <i>Description.term</i> is case insensitive and can be can be changed from upper to lower case (or vice-versa) if appropriate to the context in which it is used.

7.4.1.5 Concept enumerations for characteristicTypeld



This *concept* enumeration represents a value that can be applied to the *Relationship.characteristicTypeld* field. This is used to indicate whether a *Relationship* forms part of the definition of the source *concept*.

Table 191 shows the current set of values for this *concept* enumeration. Note that two the values 90000000000000010007 | Stated relationship (core metadata concept) | and 90000000000000011006 | Inferred relationship (core metadata concept) | are subtypes of the more general value 9000000000000006009 | Defining relationship (core metadata concept) |.

Table 191: Characteristic type (core metadata concept) (900000000000449001)

Id	Term	Comment
9000000000000006009	Defining relationship (core metadata concept)	The <i>Relationship</i> is part of the <i>description logic</i> definition of the source <i>concept</i> .
- 90000000000000010007	Stated Relationship (core metadata concept)	Indicates that this defining <i>Relationship</i> was stated by a terminology author.
- 90000000000000011006	Inferred Relationship (core metadata concept)	Indicates that this defining <i>Relationship</i> was inferred by a <i>description logic classifier</i> from the set of stated <i>Relationships</i> .

Id	Term	Comment
9000000000000225001	Qualifying relationship (core metadata concept)	The <i>Relationship</i> is not part of the definition of the <i>concept</i> but indicates a possible qualification that may be applied to refine a <i>postcoordinated expression</i> that refers to the source <i>concept</i> .
9000000000000227009	Additional relationship (core metadata concept)	The <i>Relationship</i> is not part of the definition of the <i>concept</i> but is used to convey some additional information about the <i>concept</i> . This additional information may only be applicable to a particular jurisdiction or use case.

7.4.1.6 Concept enumerations for modifierId



This *concept* enumeration represents a value that can be applied to the *Relationship.modifierId* field. This is used to indicate the type of *Description Logic* (DL) restriction (some, all, etc.) that applies to the *Relationship*.

[Table 192](#) shows the current set of values for this *concept* enumeration.

Table 192: Modifier (core metadata concept) (900000000000450001)

Id	Term	Comment
900000000000451002	Existential restriction modifier (core metadata concept)	Indicates that <i>description logic</i> restriction represented by this defining <i>Relationship</i> applies to some aspect of the <i>concept</i> .
900000000000452009	Universal restriction modifier (core metadata concept)	Indicates that <i>description logic</i> restriction represented by this defining <i>Relationship</i> applies to all aspects of the <i>concept</i> .

7.4.1.7 Concept enumerations for identifierSchemeld



This *concept* enumeration represents a value that can be applied to the *Identifier.identifierSchemeld* field. This is used to indicate the scheme to which the *Identifier* value belongs.

[Table 193](#) shows the current set of values for this *concept* enumeration. This set of values is extensible to allow additional *Identifiers* to be used to represent *SNOMED CT components* where this is necessary.

Table 193: Identifier scheme (core metadata concept) (900000000000453004)

Id	Term	Comment
90000000000002006	SNOMED CT universally unique identifier (core metadata concept)	The identification scheme in which the <i>Identifiers</i> are <i>UUID</i> 's allocated to <i>SNOMED CT components</i> .

Id	Term	Comment
900000000000294009	SNOMED CT integer identifier (core metadata concept)	The scheme comprising all <i>SNOMED Clinical Terms Identifiers (SCTID)</i> .

7.4.1.8 Other Concept enumerations



Reference sets can also include *concept* enumeration values and the values for these are *subtypes* of 900000000000491004 | Attribute value (foundation metadata concept) |. The values applicable to each *Attribute* in each type of *Reference set* are specified by the 900000000000456007 | Reference set descriptor reference set (foundation metadata concept) |.

👉 **Note:** In the current pre-release RF2 data some sets of *concept* enumerations are *subtypes* of 900000000000457003 | Reference set attribute (foundation metadata concept) |. However, in future it is anticipated that they will all be *subtypes* of 900000000000491004 | Attribute value (foundation metadata concept) |.

7.4.2 Essential Reference Sets



The *Reference Set* mechanism provides flexibility and extensibility to the core terminology. The *Reference Sets* described in this section are essential and need to be supported by all *SNOMED CT enabled terminology servers*.

Other *Reference Sets* are used to deliver specific added value functionality and/or for local configuration. While implementers are advised to consider providing full *Reference Set* support the specific requirements for these depend on the intended uses of the systems and these are described elsewhere in the guide.

7.4.2.1 Language Reference Sets



At least one language *Reference Set* needs to be imported. This is essential to enable the *preferred term* to be identified for each *concept*.

The language *Refsets* supported in the *International Release* are shown in [Table 194](#).

Table 194: English [International Organization for Standardization 639-1 code en] language reference set (foundation metadata concept) (900000000000507009)

Id	Term	Comment
900000000000508004	Great Britain English language reference set (foundation metadata concept)	
900000000000509007	United States of America English language reference set (foundation metadata concept)	

The Language *Reference Set* hierarchy is extensible and other languages and *dialects* will be added to the hierarchy shown in [Figure 93](#) to either as part of the *International Release* or an *Extension*.

- 900000000000506000 | Language type reference set (foundation metadata concept) |
 - 900000000000507009 | English [International Organization for Standardization 639-1 code en] language reference set (foundation metadata concept) |

- 900000000000508004 | Great Britain English language reference set (foundation metadata concept)
 - |
- 900000000000509007 | United States of America English language reference set (foundation metadata concept) |

Figure 93: The Language Reference Set hierarchy

Each language *Reference set* refers to each of the *Descriptions* that is used in that language or *dialect* and assigns a value for the acceptability of the term associated with that *Description* when applied to the *Concept* associated with that *Description*. The values for acceptability are *concept* enumerations show in [Table 195](#).

Table 195: Acceptability (foundation metadata concept) (900000000000511003)

Id	Term	Comment
900000000000548007	Preferred (foundation metadata concept)	<p>The term associated with this <i>description</i> is the preferred <i>description</i>, of the specified <i>Description.type</i>, for the associated <i>concept</i>, in the language or <i>dialect</i> represented by this <i>Reference set</i>.</p> <ul style="list-style-type: none"> • If the <i>Description.type</i> is <i>synonym</i>, this <i>description</i> is the <i>preferred term</i>. • If the <i>Description.type</i> is <i>fully specified name</i> this <i>description</i> is the preferred <i>fully specified name</i>. <p>For each <i>concept</i> there should be exactly one preferred <i>description</i> of each <i>Description.type</i> in each language <i>Reference set</i>.</p>
900000000000549004	Acceptable (foundation metadata concept)	<p>The term associated with this <i>description</i> is acceptable for use in language or <i>dialect</i> represented by this <i>Reference set</i>.</p> <p>For each <i>concept</i> there may be any number of acceptable <i>descriptions</i> of each <i>Description.type</i> in each language <i>Reference set</i>.</p>

7.4.2.2 Component Inactivation Reference Sets



The *Component Inactivation Reference Sets* are required to determine the reason why a *concept*, *Description* or *Relationship* is inactive. The *boolean active* field in each *component* indicates whether it is active but does not explain why a previously active *component* has been inactivated. The reason for inactivation may affect the way in which *components* that have been made inactive are dealt with when they have been used to create records, protocols or queries prior to inactivation.

The three *Component Inactivation Reference Sets* are shown in [Table 196](#).

Table 196: Component Inactivation Reference Sets

Id	Fully Specified Name	Note
9000000000489007	Concept inactivation indicator attribute value reference set (foundation metadata concept)	Indicates the reason that a <i>concept</i> has been made inactive.
9000000000489008	Description inactivation indicator attribute value reference set (foundation metadata concept)	Indicates the reason that a <i>Description</i> has been made inactive.
9000000000547002	Relationship inactivation indicator attribute value reference set (foundation metadata concept)	(Not currently provided - for future use)

The reason for inactivation is specified by a *concept* enumeration. The permitted values for this enumeration for a *Concept* are shown in [Table 197](#) and the permitted values for a *Description* are shown in [Table 198](#).

Table 197: Concept inactivation value (foundation metadata concept) (900000000000481005)

Id	Fully Specified Name	Comment
9000000000482003	Duplicate component (foundation metadata concept)	The <i>Concept</i> has been made inactive because it has the same meaning as another <i>Concept</i> .
9000000000483008	Outdated component (foundation metadata concept)	The <i>Concept</i> has been made inactive because it is an outdated <i>concept</i> that is no longer used.
9000000000484002	Ambiguous component (foundation metadata concept)	The <i>Concept</i> has been made inactive because it is inherently ambiguous either because of an incomplete <i>fully specified name</i> or because it has several associated terms that are not regarded as synonymous or partial synonymous.
9000000000485001	Erroneous component (foundation metadata concept)	The <i>Concept</i> has been made inactive because it contains an error.
9000000000486000	Limited component (foundation metadata concept)	The <i>Concept</i> is of limited value as it contains classification categories such as 'Not Elsewhere Classified' which do not have a stable meaning within <i>SNOMED CT</i> . Until 2010 <i>concepts</i> with this status were regarded as active but since then they have been marked as inactive.
9000000000487009	Component moved elsewhere (foundation metadata concept)	The <i>Concept</i> has been made inactive because it has been moved to another namespace.

Id	Fully Specified Name	Comment
90000000049206	Pending move (foundation metadata concept)	The <i>Concept</i> is still active but it is in the process of being moved to another namespace and when the move is complete it will be marked as inactive.

Table 198: Description inactivation value (foundation metadata concept) (9000000000000493001)

Id	Fully Specified Name	Comment
90000000049203	Duplicate component (foundation metadata concept)	The <i>Description</i> has been made inactive because it duplicates another <i>Description</i> .
90000000049308	Outdated component (foundation metadata concept)	The <i>Description</i> has been made inactive because it is an outdated name or spelling that is no longer used.
90000000049501	Erroneous component (foundation metadata concept)	The <i>Description</i> has been made inactive because it contains an error.
90000000049600	Limited component (foundation metadata concept)	The <i>Description</i> refers to a <i>Concept</i> that has limited status. 👉 Note: This value should not be used in future releases as Limited status <i>Concepts</i> are now inactive. However, this value may appear on retrospective data in a <i>full release</i> .
90000000049709	Component moved elsewhere (foundation metadata concept)	The <i>Description</i> has been made inactive because it has been moved to another namespace.
90000000049206	Pending move (foundation metadata concept)	The <i>Description</i> is still active but it is in the process of being moved to another namespace and when the move is complete it will be marked as Inactive.
90000000049407	Inappropriate component (foundation metadata concept)	The <i>Description</i> has been made inactive because the associated term does not describe the associated <i>Concept</i> .
90000000049508	Concept non-current (foundation metadata concept)	The <i>Description</i> is still active but the <i>Concept</i> it refers to is now inactive.

The *component.active* field allows rapid determination of whether a *component* is intended for active use. However, where a full interpretation of the status of a *component* is required two factors must be taken into account. The absence of a row in the relevant inactivation *Refset* implies a default meaning which and this default meaning depends on whether the *component* is active or inactive:

- For an active *component* it means active and in current use as distinct from active *Pending move*
- For an *inactive component* it means inactive with no reason given for inactivation.

This leads to the set of interpretations for each possible combination of values shown in [Table 199](#).

Table 199: Concept Status evaluation table

Most recent <i>Concept</i> row for a <i>concept.id</i>	Most recent <i>Refset</i> row for the <i>RefsetMember.id</i> in “ <i>Concept inactivation Refset</i> ” for the <i>concept.id</i>		<i>ConceptStatus</i> (with RF1 enumerated value)
Exists/active	Exists/active	<i>valueId</i>	
None	None	-	<i>Not applicable (not yet released)</i>
Active	None or Inactive	-	Current (0)
Inactive	None or Inactive	-	Inactive - no reason given (1)
Inactive	Active	900000000000482003	duplicate (2)
Inactive	Active	900000000000483008	outdated (3)
Inactive	Active	900000000000484002	ambiguous (4)
Inactive	Active	900000000000485001	erroneous (5)
Inactive	Active	900000000000486000	limited (6) (from 2010-01-31)
Active	Active	900000000000486000	limited (6) (before 2010-01-31)
Inactive	Active	900000000000487009	moved elsewhere
Active	Active	900000000000492006	pending move
Any combinations not shown above		<i>Future releases may add new values or rules. Otherwise, values and combinations not shown in this table have no agreed interpretation and must be regarded as data errors.</i>	

7.4.2.3 Historical Association Reference Sets



Historical Association Reference Sets provide links between *inactive concepts* and their active replacements or equivalents. There is one Historical Association Reference Set for each type of historical association as shown in [Table 200](#).

Table 200: Historical association reference set (foundation metadata concept) (900000000000522004)

Id	Term	Comment
900000000000523009	POSSIBLY EQUIVALENT TO association reference set (foundation metadata concept)	Applies to a <i>concept</i> that is ambiguous. The targetComponent is an active <i>concept</i> that represents one of the possible meanings of the inactive <i>concept</i> . Multiple rows are used to refer to each of the possible meanings of the ambiguous <i>concept</i> . Previously referred to as "MAY BE A".
900000000000524003	MOVED TO association reference set (foundation metadata concept)	Applies to a <i>component</i> that has been moved to (or are pending a move to) another namespace. The targetComponent identifies the target namespace (not the new <i>component</i>).
900000000000525002	MOVED FROM association reference set (foundation metadata concept)	Applies to a <i>component</i> that has been moved to this namespace from another namespace. The targetComponent identifies the original <i>component Identifier</i> in its previous namespace.
900000000000526001	REPLACED BY association reference set (foundation metadata concept)	Applies to an erroneous, obsolete and other <i>inactive component</i> for which there is a single active replacement. The targetComponent identifies the active <i>component</i> that replaces this <i>component</i> .
900000000000527005	SAME AS association reference set (foundation metadata concept)	Applies to a <i>component</i> that is a duplicate. The targetComponent identifies the active <i>component</i> that this <i>component</i> duplicates.
900000000000528000	WAS A association reference set (foundation metadata concept)	(current usage unclear)
900000000000529008	SIMILAR TO association reference set (foundation metadata concept)	(not used currently)
900000000000530003	ALTERNATIVE association reference set (foundation metadata concept)	(not used currently)
900000000000531004	REFERS TO concept association reference set (foundation metadata concept)	(not used currently)

7.4.2.4 Module Dependency Reference Set



The *Module Dependency Reference Set* provides information about dependencies between different versions of particular development modules. This *Reference Set* (identified as 9000000000000534007 | *Module dependency reference set (foundation metadata concept)* |) should be checked when importing data to ensure that all dependencies are satisfied.

The rows in this *Reference Set* that originate in a given module (identified by *moduleId*) indicate a dependency on the module identified by the *referencedComponentId*. The two *string* values each contain dates that indicate the version of source module and the required version of the module on which it depends.

7.4.3 Optional Reference Sets



The *Reference Sets* described in the following sections are required for specific purposes. If an implementation does not need to address a particular requirement (e.g. mapping from a legacy coding scheme) or supports a more up to date approach (e.g. the *Machine Readable Concept Model* rather than the use of *refinability* flags) then that *Reference Set* need not be imported or may be imported and not used.

7.4.3.1 Relationship Refinability Reference Set



The *Relationship Refinability Reference Set* provides information about whether it is permissible to refine the value of a *Relationship*. This *Reference Set* is identified as 900000000000488004 | *Relationship refinability attribute value reference set (foundation metadata concept)* | and its *Concept enumeration* values are specified in [Table 201](#).

Table 201: Refinability value (foundation metadata concept) (900000000000226000)

Id	Term	Comment
9000000000000007000	<i>Not refinable (foundation metadata concept)</i>	The value provided by the <i>destinationId</i> may be used but none of the <i>subtypes</i> of this <i>concept</i> are permitted.
900000000000218008	<i>Mandatory refinability (foundation metadata concept)</i>	The value may be refined by selecting a <i>subtype</i> of the <i>concept</i> referred to by the <i>destinationId</i> .
900000000000216007	<i>Optional refinability (foundation metadata concept)</i>	The value may be refined by selecting a <i>subtype</i> of the <i>concept</i> referred to by the <i>destinationId</i> .

👉 **Note:** This information is equivalent to the *Release Format 1 Relationships*. *refinability* field. Its value is likely to diminish over time as the *Machine Readable Concept Model* provides a more complete representation of *refinability*.

7.4.3.2 Legacy Code Map Reference Sets



Legacy Code Map Reference Sets are simple maps to *SNOMED CT* from legacy code systems, including *SNOMED* codes (i.e. codes used in *SNOMED 3*) and *NHS Clinical Terms Version 3 Identifiers* (including all versions of the *Read Codes*). There is one *Reference Set* for legacy *SNOMED* codes and one for *Clinical Terms Version 3* as shown in [Table 202](#).

In both cases, the *referenceComponentId* refers to a *SNOMED CT concept* and the *mapTarget* *string* value is the code in the other coding scheme.

Table 202: Simple map type reference set (foundation metadata concept) (900000000000496009)

Id	Term	Comment
900000000000497000	CTV3 simple map reference set (foundation metadata concept)	The map between <i>Clinical Terms Version 3</i> and all version of the <i>Read Codes</i> and <i>SNOMED CT</i> .
900000000000498005	SNOMED RT identifier simple map (foundation metadata concept)	The map between legacy <i>SNOMED</i> codes and <i>SNOMED CT</i> .

7.4.4 Reference Sets supporting advanced functionality



Some of the *Reference Sets* included as part of the *SNOMED CT International Release* support advanced uses and may not need to be implemented. In particular *Reference Sets* that provide information about other *Reference Set* can be valuable but are not essential provided the implementation fully supports all the *Reference Sets* required by its users.

7.4.4.1 Description Format Reference Set



The *Description Format Reference Set* provides information about the format of each of the *Description* types. This *Reference Set* is identified as 900000000000538005 | Description format reference set (foundation metadata concept) |.

The *referencedComponentId* of each member of the *reference set* refers to a *Description* type, represented by a *subtype* of the concept 900000000000446008 | Description type (core metadata concept) |. The *descriptionFormat* refers to one of the *Concept enumeration* values shown in [Table 203](#). The *descriptionLength* indicates the longest permitted string for this *Description* type.

Table 203: Description format (foundation metadata concept) (900000000000539002)

Id	Term	Comment
900000000000540000	Plain text (foundation metadata concept)	<i>Descriptions</i> of this types linked to this format are in plain text. This applies <i>fully specified names</i> and <i>synonyms</i> .
900000000000541001	Limited HyperText Markup Language (foundation metadata concept)	<i>Descriptions</i> of this types linked to this format use a limited version of HTML markup.
900000000000542008	Extensible HyperText Markup Language (foundation metadata concept)	<i>Descriptions</i> of this types linked to this format may use the full scope of XHTML markup.
900000000000543003	Darwin Information Typing Architecture (foundation metadata concept)	<i>Descriptions</i> of this types linked to this format are represented as <i>DITA</i> topics using XML markup.

7.4.4.2 Reference Set Descriptor Reference Set



The *Reference Set Descriptor Reference Set*, which is identified as 900000000000456007 | *Reference set descriptor reference set (foundation metadata concept)* |, provides information about the structure of each type of *Reference Set*.

The first six fields of each *Reference Set* have the same structure but additional attributes can be included to meet specific requirements. The *Reference Set Descriptor Reference Set* provides a machine readable representation that can be used to allocate and locate appropriate storage for each type of *Reference Set*.

7.4.5 Using other Reference Sets



7.4.5.1 Importing Reference Sets



One or more *Reference Sets* may be held in a single *Reference Set release file*. However, if there are more than one *Reference Sets* in a single file, they will all have the same structure (i.e. - the same number of additional fields of the same top level types of *component*, *Integer* or *String*).

Each record in the *Reference Set* file represents a member of the *reference set*. The *refsetId* column identifies the *Reference Set* that the member record belongs to.

The *refsetId* is an *SCTID* that can be used to look up the *concept* in the | *Reference Set* | metadata that describes the *reference set*. Up to three *Descriptions* (with three different *typeid*s) may be associated with the *Reference Set concept*:

- A *Description* with a *typeid* of |FSN|, used to formally describe the *Reference Set*. This *Description* will always exist.
- A *Description* with a *typeid* of | Synonym |, used to name the *Reference Set*. This *Description* will always exist, and can be used to display the name of the *Reference Set* within a system.
- A *Description* with a *typeid* of |Purpose|, used to describe the purpose of the *Reference Set*. This *Description* may or may not be present.

The *refsetId* can also be used to look up the *Reference Set Descriptor*, in the | *Reference set descriptor* | *Reference Set*. This can be done by identifying the member records in the | *Reference set descriptor* | *reference set* with a *referencedComponentId* that matches the *refsetId* of the *Reference Set*.

There will be one *Descriptor* record describing the *referencedComponentId* field in the *Reference Set* and one additional record for each optional field within the *Reference Set*. The *Descriptor* record with an *attributeOrder* field value of '0' describes the *referencedComponentId* field; a *Descriptor* record with an *attributeOrder* field value of '1' would describe the first optional field; etc.

For each *Reference set* field being described (i.e. - the *referencedComponentId* and each optional field), two fields in the *Descriptor* record provide additional information:

- The *attributeType* field is a reference to a *concept* under the | *Attribute type* | metadata *hierarchy* that provides typing information for the field. At the top level, this could be | *component type* |, | *Integer* | or | *String* |, and would then match the typing information available within the *Reference Set* file name (see the [SNOMED CT - File Naming Conventions](#)). However, the type of a field can also be specified at a finer level of granularity using the *attributeType* field. For instance, instead of the *attributeType* being specified simply as an | *Integer* |, it may instead be specified as an | *Unsigned integer* | or a | *Signed integer* |. For a full list of types, see the | *Attribute type* | metadata *hierarchy*.
- The *attributeDescription* field is a reference to a *concept* under the | *Reference set attribute* | metadata *hierarchy* that also provides additional information about each *Reference Set* field. Up to three *Descriptions* (with three different *typeid*s) may be associated with each of these *concepts*:
 - A *Description* with a *typeid* of |FSN|, used to formally describe the *Reference Set* field. This *Description* will always exist.

- A *Description* with a *typeid* of |*Synonym*|, used to name the *Reference Set* field. This *Description* will always exist, and can be used to display a column header for each *Reference Set* field used within a system.
- A *Description* with a *typeid* of |*Purpose*|, used to describe the purpose of the *Reference Set* field. This *Description* may or may not be present.

Additionally, if the *attributeType* is |*Concept* type component|, then the *children* of the *concept* referred to by the *attributeDescription* provide a list of allowed *concept* enumeration values for the *Reference Set* field. Each of these *concepts* will have two *Descriptions* with *typeid*s of |*FSN*| and of |*Synonym*|, and the latter set of *Descriptions* can be used to validate field entry for *concept* enumeration type *Reference Set* fields or to create pick-lists to allow users to select one or more values. Where the *attributeDescription* *concept* does not have any *children*, then no limitation is placed on the *concepts* allowed in the *Reference Set* field.

7.4.5.2 Using Reference Sets without Descriptors



All *Reference Sets* that are released from *IHTSDO* or from a *National Release Center* will have an associated Descriptor for the *Reference Set*. However, Descriptors are optional for other organizations that create *Reference Sets*. Where you are using a *Reference Set* for which a Descriptor has not been created, and you need additional information about the *Reference Set*, the Descriptor of the closest *ancestor* of the *concept* describing the *Reference Set* that does have a Descriptor may be used. This situation should be rare, as an organization that releases *Reference Sets* should only release them without Descriptors if it is sure that its consumers do not require the information held within the Descriptors.

7.4.5.3 Using Reference Sets to hold simple value sets



Where it is known that a single simple *Reference Set* is held in a file, a simple *value set* may be retrieved from the *Reference Set* by taking the *referencedComponentIds* of each record with an *active* field set to '1'. Each value in the *value set* is then an *SCTID* of a *SNOMED CT component*.

Where a *release file* contains multiple simple *Reference Sets*, then a number of *value sets* may be retrieved from the file by taking the *referencedComponentIds* of each record with an *active* field set to '1', and grouping them into *value sets* by using the *refsetId* field. Each value in the *value set* is an *SCTID* of a *SNOMED CT component*. In order to retrieve the name of each *value set*, its *refsetId* can be used to identify a |*Reference set*| metadata *concept* that will have a *Description* with a *typeid* of |*Synonym*| that provides a name for the *value set*.

7.5 Foundation Terminology services



This section summarizes a set of services that all *terminology servers* require. Some of these services are described in more detail in subsequent sections. The more advanced services specified in other sections depend on one or more of these foundation services.

7.5.1 Access to release information



Terminology servers should enable client applications and users to access the *current SNOMED CT release version information*.

7.5.2 Access to components



Most *Terminology services* depend on the ability to efficiently access information about the set of components in a selected *snapshot view*. The following sections outline the types of information that need to be accessible and provide illustration of some of the common patterns of data access that are required. The illustrations are expressed as SQL queries based on the *example relational representation* and *dynamic snapshot views approaches* discussed in earlier sections.

7.5.2.1 Access to concepts



A *terminology server* should enable client applications to rapidly find the current version of a *Concept* by its unique *Identifier* (*Concept.id*).

Once a *Concept* has been found, the client application should be able to read the values of the properties of that *Concept* which are either:

1. Provided directly as *concept file* fields:

- active;
- *definitionStatusId*.

2. Provided indirectly through associations to other *components*:

- *Descriptions*.
- *Relationships*.

3. Provided indirectly via relevant *Reference sets*:

- For example *Information about Inactive Concepts*.

7.5.2.1.1 Information about Inactive Concepts



The *Concept.active* field is a *boolean* value which distinguishes between active and *inactive* *concepts*. To find out more information about the status of a *concept* it is necessary to look for a relevant row in the 900000000000480006 | Attribute value type reference set (foundation metadata concept) |.

The example query below illustrates this process.

```
/* sv_concept refers to a snapshot view of concept */
/* sv_refset_status refers to a snapshot view of the */
/* inactivation Refset with term lookup see below */

SELECT `c`.`id` AS `ConceptId`,
(CASE WHEN (`r`.`RsActive` = 1) THEN
`r`.`ValueTerm` else
(CASE WHEN `c`.`active` THEN
'Current' ELSE
'Inactive no reason' END)
END) AS `Status`
FROM (`sv_concept` `c`
LEFT JOIN `sv_refset_status` `r`
ON (`r`.`ItemId` = `c`.`id`))
WHERE `c`.`id`=[some-concept-id];

/* Query generating the sv_refset_status view */
/* sv_refset_c is snapshot view of the cRefset table */
/* sv_fsn is a snapshot view of fully specified names */
/* See section on access to Descriptions for details */

SELECT `r`.`active` AS `RsActive`, `r`.`referencedComponentId` AS `ItemId`,
`d2`.`conceptId` AS `ValueId`, `d2`.`term` AS `ValueTerm`
FROM (`sv_refset_c` `r` join `sv_fsn` `d2`)
WHERE ((`d2`.`conceptId` = `r`.`valueId`)
AND (`r`.`refsetId` = 900000000000489007));
```

Figure 94: Determining concept status

If a *concept* is inactive then, it may be necessary to follow the historical associations to locate the *active concept(s)* that have replaced or disambiguated the *inactive concept*. [Figure 95](#) illustrates and finds the id of the active equivalent of a duplicate *concept*.

```
/* Find SAME AS reference for a duplicate concept */
/* sv_refset_c is snapshot view of the cRefset table */


```

```
SELECT `targetComponent`
  FROM `sv_refset_c`
 WHERE `refsetId` = 900000000000527005
   AND `referencedComponentId` = [some-concept-id];
```

Figure 95: Following historical associations

7.5.2.2 Access to Descriptions



A terminology server should enable client applications to rapidly find the current version of any *Description* or set of *Descriptions* by any of the following criteria:

- Its unique *Identifier* (*Description.id*);
- *conceptId* of the *concept* with which it is associated;
- A combination of *conceptId*, *DescriptionType*, Language or *dialect* and Acceptability (in that language or *dialect*).

Once a *Description* has been found the client application should be able to read the values of any of the properties of that *Description* which are either:

- Provided directly as *Description file* fields:
 - *active*;
 - *term*;
 - *caseSignificanceId*
 - *languageCode*;
 - *typeId* (the *Description Type*).
- Provided indirectly via relevant *Reference sets*:
 - For an example see [Determining Description Type and Acceptability](#).

7.5.2.2.1 Determining Description Type and Acceptability



The *active* field indicates whether the *Description* is in current active use. The *typeId* and *languageCode* indicate the *Description type* and the language of the associated term. This information is useful but it is not sufficient to determine the *preferred term*. In order to determine the acceptability of or preference for use of a particular *Description* it is necessary to apply a language *Reference set*. This is illustrated by [Figure 96](#).

```
/* sv_description is a snapshot view of the description file */
/* sv_refset_c is snapshot view of the cRefset table */
/* configLang() is a function that returns the chosen language RefsetId */

SELECT `d`.*
  FROM (`sv_description` `d` JOIN `sv_refset_c` `rs`
    ON(`d`.`id` = `rs`.`referencedComponentId`))
   WHERE ((`d`.`active` = 1) AND (`d`.`typeId` = 90000000000013009)
     AND (`d`.`conceptId` = [some-concept-id]) AND (`rs`.`refsetId` = `configLangId`()))
     AND (`rs`.`active` = 1) AND (`rs`.`valueId` = 900000000000548007));
```

Figure 96: Identifying the preferred term

The *Fully Specified Name* for a particular language or *dialect* can also be determined in the same way as shown in [Figure 97](#). The only difference between this and the *preferred term* example is the change in the *typeId* predicate.

 **Note:** The *Fully Specified Name* may not be present in all supported languages therefore a fall-back to the US English may be necessary.

```
SELECT `d`.*  
FROM (`sv_description` `d` join `sv_refset_c` `rs`  
    ON(`d`.`id` = `rs`.`referencedComponentId`))  
    WHERE ((`d`.`active` = 1) AND (`d`.`typeId` = 90000000000000003001)  
        AND (`d`.`conceptId`=[some-concept-id] AND (`rs`.`refsetId` = `configLangId`())  
        AND (`rs`.`active` = 1) AND (`rs`.`valueId` = 900000000000548007)));
```

Figure 97: Identifying the preferred fully specified name

[Figure 98](#) illustrates an approach to returning all the acceptable or *preferred terms* together with an indication of which *Description* type and preference.

```
/* sv_description is a snapshot view of the description file */  
/* sv_refset_c is snapshot view of the cRefset table */  
/* configLang() is a function that return the chosen language RefsetId */  
  
SELECT `d`.*, (CASE WHEN `rs`.`valueId`=900000000000548007 THEN  
    'Preferred' ELSE  
    'Acceptable' END) AS `Acceptability`  
    (CASE WHEN `d`.`typeId`=90000000000013009 THEN  
    'Synonym' ELSE  
    'FSN' END) AS `DescriptionType`  
FROM (`sv_description` `d` join `sv_refset_c` `rs`  
    ON(`d`.`id` = `rs`.`referencedComponentId`))  
    WHERE ((`d`.`active` = 1) AND ((`d`.`typeId` = 90000000000013009) OR (`d`.`typeId` =  
90000000000000003001))  
        AND (`d`.`conceptId`=[some-concept-id] AND (`rs`.`refsetId` = `configLangId`())  
        AND (`rs`.`active` = 1) AND  
        ((`rs`.`valueId` = 900000000000548007) OR (`rs`.`valueId` = 900000000000548007)));
```

Figure 98: Finding all the acceptable terms

7.5.2.3 Access to Relationships



A *terminology server* should enable a client application to rapidly find the current version of any *Relationship* or set of *Relationships* by any of the following criteria:

- Its unique *Identifier Relationship.id*;
- *sourceld*
- *sourceld, characteristicTypeld* and *typeld*
- *sourceld, characteristicTypeld, relationshipGroup* and *typeld*
- *destinationId*
- *destinationId, characteristicTypeld* and *typeld*

Once a *Relationship* has been found the client application should be able to read the values of any of the properties of that *Relationship*:

- Provided directly as *Relationship file* fields:
 - *active*;
 - *sourceld*
 - *characteristicTypeld*
 - *typeld*
 - *destinationId*

- *relationshipGroup*
- *modifierId*
- Provided indirectly in the *concepts* that it refers to:
 - For example [Using and traversing relationships](#).

7.5.2.3.1 Using and traversing Relationships



The defining *Relationships* of a *concept* can be shown by following the relevant *concept identifier* and displaying the relevant terms as showing in [Figure 99](#).

```
/* sv_relationship is a snapshot view of the relationship file */
/* sv_pref is a snapshot of descriptions filtered to preferred term */

SELECT `r`.`typeld` AS `type_id`, `typ`.`term` AS `type_term`
, `r`.`destinationId` AS `dest_id`, `dest`.`term` AS `dest_term`
, `r`.`relationshipGroup` AS `relationshipGroup`
FROM (((`sv_relationship` `r`
JOIN `sv_pref` `src`
ON ((`r`.`sourceld` = `src`.`conceptId`)))
JOIN `sv_pref` `typ`
ON ((`r`.`typeld` = `typ`.`conceptId`)))
JOIN `sv_pref` `dest`
ON ((`r`.`destinationId` = `dest`.`conceptId`)))
WHERE ((`r`.`active` = 1)
AND (`r`.`characteristicTypeld` = 90000000000000006009)
AND (`r`.`sourceld` = [some-concept-id]));
```

Figure 99: Showing the defining Relationships of a concept

A simplification of the defining *Relationship* query can be used to return the *supertype parent concepts* as shown in [Figure 100](#).

```
/* sv_relationship is a snapshot view of the relationship file */
/* sv_pref is a snapshot of descriptions filtered to preferred term */

SELECT `r`.`destinationId` AS `id`, `d`.`term` AS `term`, `r`.`sourceld` AS `conceptId`
FROM (`sv_relationship` `r`
JOIN `sv_pref` `d`
ON ((`r`.`destinationId` = `d`.`conceptId`)))
WHERE ((`r`.`active` = 1)
AND (`r`.`typeld` = 116680003)
AND (`r`.`sourceld` = [some-concept-id]));
```

Figure 100: Showing the supertype parents of a concept

By swapping the *sourceld* and *destinationId* from the previous example the *subtype children* of the *concept* can be displayed as shown in [Figure 101](#).

```
/* sv_relationship is a snapshot view of the relationship file */
/* sv_pref is a snapshot of descriptions filtered to preferred term */

SELECT `r`.`sourceld` AS `id`, `d`.`term` AS `term`, `r`.`destinationId` AS `conceptId`
FROM (`sv_relationship` `r`
JOIN `sv_pref` `d`
ON ((`r`.`sourceld` = `d`.`conceptId`)))
WHERE ((`r`.`active` = 1)
AND (`r`.`typeld` = 116680003)
AND (`r`.`destinationId` = [some-concept-id]));
```

Figure 101: Showing the subtype children of a concept

7.5.3 Access to essential concept Identifiers



Terminology servers should provide efficient access to the *Identifiers* that represent *concepts* with structurally significant *Roles* within the terminology. [Table 204](#) lists the *concepts* that have the most clear-cut structurally significant *Roles*. A terminology server should enable access to these *Identifiers* by an easy to use name of enumeration. In addition a terminology server should provide a service that rapidly determines whether a given *concept* is a *subtype* of any of these *concepts*. It is also useful for the terminology server to extend similar functionality to all direct *subtypes* of the *root concept* (| SNOMED CT Concept |) and to *subtypes* descendants of | concept model attribute |.

Table 204: Essential concept Identifiers

Id	Preferred Term	Significance
138875005	SNOMED CT Concept	The <i>root concept</i> . All other <i>active concepts</i> are <i>subtypes</i> of this <i>concept</i> .
9000000000000441003	SNOMED CT Model Component	All active metadata <i>concepts</i> are <i>subtypes</i> of this <i>concept</i> .
9000000000000442005	core metadata concept	All enumerated values applicable to core <i>components</i> are <i>subtypes</i> of this <i>concept</i> .
9000000000000454005	foundation metadata concept	All <i>reference sets</i> and all <i>reference set related metadata concept</i> are <i>subtypes</i> of this <i>concept</i> .
9000000000000455006	reference set	All <i>reference sets</i> are <i>subtypes</i> of this <i>concept</i> .
116680003	is a	The <i>Attribute</i> used to specify the <i>subtype Relationship</i> between <i>concepts</i> .
246061005	attribute	All <i>Attribute (relationship type) concepts</i> are <i>subtypes</i> of this <i>concept</i> .
410662002	concept model attribute	With the exception of the <i>subtype Relationship</i> (see above) all <i>relationship types</i> that are used in the <i>SNOMED CT Concept Model</i> are <i>subtypes</i> of this <i>concept</i> .
370136006	namespace concept	Each <i>subtype</i> of this <i>concept</i> represents an extension namespaces allocated by the IHTSDO.

Id	Preferred Term	Significance
363743006	navigational concept	Subtypes of this concept to provide nodes in <i>navigation hierarchies</i> . They act as grouper categories that do not have any semantic meaning and thus do not appear elsewhere in the <i>SNOMED CT</i> hierarchy.

7.6 User Interface Terminology services



This section of the guide is concerned with *Terminology services* that allow users to view and select of *SNOMED CT Concepts* and *Descriptions*.

7.6.1 Text Searches



Effective implementation of *SNOMED CT* depends on the speed and simplicity with which users can locate the *terms* and *concepts* that they wish to use. A busy clinical user may become frustrated if the content they need cannot be quickly located when they search using familiar words or phrases. For this reason an efficient search strategy should address the following issues:

- Speed of search:
 - Search speed should be optimized by use of appropriate indexes.
- Search should not be too sensitive to word *order* or exact phrasing:
 - Search should be insensitive to word - *order* variants:
 - For example, "head pain" for | pain in head |
 - Allow use of acronyms or abbreviations for frequently used *terms*:
 - For example, "MI" for "myocardial infarction" or "mitral incompetence".
 - Search should take account of word form variants:
 - For example, "inflamed", "inflammatory", "inflammation".
- Excessive search results should not hinder selection of the required *concept*.
 - When several *synonyms* of the same *concept* match the search key, only one should be displayed.

The purpose of this section of the implementation guide is to describe strategies a developer might use to implement the search requirements outlined above.

The *SNOMED CT* Developer Toolkit contains several files, which help to support efficient search mechanisms. These include the *Excluded Words Table*, four *keyword* indexes and the *Word Equivalents Table* summarized by [Table 31](#) and [Table 32](#).

7.6.1.1 Single keyword index



The single *keyword* table, (*DescWordKey*), provides a pointer from each *keyword* used in any *Description*, to the *Descriptions* in which that *keyword* is used. The purpose of the single *keyword* index is to support a search capability, which is independent of the *order* in which words appear in a *description*. The single *keyword* index represents the minimum necessary supporting structure for searches on *SNOMED CT*.

content. Searches involving target words that appear in many *descriptions* may be unacceptably slow if searches are carried out using the single *keyword* index alone. Developers wishing to produce applications with faster search times are encouraged to supplement their system with a multiple *keyword* index such the DescDualkey table (see [Word Search Tables](#)) provided as part of the *SNOMED CT release*.

Note that some words that are used in *description* are linking words, which are unlikely to be in the target of a search. These words are not considered to be *keywords* and may be excluded from the *keyword* index. They are found in *Excluded Words File*.

7.6.1.1.1 Generating the single keyword index



Although single *keyword* indexes are available as part of the *International Release*, developers need to know how to add *keyword* entries for any locally generated *descriptions* added as part of an *Extension*.

Entries may be added to the single *keyword* table by following the method outlined below.

For each *description*, parse the text of the *term*:

- To avoid inappropriate case mismatches, convert all characters to the same case.
- Extract words by breaking at spaces, punctuation marks, and brackets.
- For each word:
 - If the word is not in a list of *excluded words*, add a row to *keyword* table.

7.6.1.1.1 Example: Generation of keywords for a sample Description



Table 205: Sample Description

Description Identifier	Concept Identifier	Term
22565018	13185000	pyrogallol 1,2-oxygenase

- Convert all characters to the same case.
| pyrogallol 1,2-oxygenase | -> "PYROGALLOL 1,2-OXYGENASE"
- Extract words by breaking at spaces, punctuation marks, and brackets.
"PYROGALLOL 1,2-OXYGENASE" -> ' (1) = "PYROGALLOL"
(2) = "1"
(3) = "2"
(4) = "OXYGENASE"
- For each word:
 - If the word is not in a list of *excluded words*, and length of word > 1, and first character is not numeric:
 - Add a row to *keyword* tables;
 - Only the first eight characters are used in the *keyword*.

Table 206: DescKey Words

KeyWord	Description Identifier
PYROGALL	22565018

KeyWord	Description Identifier
OXYGENAS	22565018

Table 207: ConcKeyWords

KeyWord	Concept Identifier
PYROGALL	13185000
OXYGENAS	13185000

7.6.1.1.2 Search using the single keyword index

A single *keyword* search may be conducted as follows:

- The user-typed search *string* is converted to consistent case;
- The *string* is parsed, breaking at spaces and punctuation characters;
- One word is selected from the parsed word list to use as a look-up on the single *keyword* index;
- Look-up on the single *keyword* index may be "exact" or "starts with," depending on wild card conventions used in the search *string*.

7.6.1.1.2.1 Example: Search using single key-word index

The user searches for "Hip* replacement*" (where "*" represents the wild card for any number of extra characters).

- The user-typed search *string* is converted to consistent case.
- "Hip* replacement" -> "HIP* REPLACEMENT*"
- The *string* is parsed, breaking at spaces and punctuation characters.
- "HIP* REPLACEMENT*" -> (1) "HIP"
- (2) "REPLACEMENT"
- Look up "HIP" on the single *keyword* index using "starts with" *query*.

Table 208: Example results for a Search for "hip"

Count	Description Identifier	Concept Identifier	Term
1	49926016	29836001	hip
2	196344018	24136001	hip
3	2296013	736004	abscess of hip
4	1480791012	386649003	partial hip replacement by prosthesis
.....

Count	Description Identifier	Concept Identifier	Term
315	371616001	1210239015	methenamine hippurate 1g tablet

Descriptions in the search results are converted to consistent case and screened, to see if they contain any words starting with "REPLACEMENT" - only those terms that do are included in the final search results.

Using a *Dual Key* index is more efficient as the same search finds only 11 matches.

Table 209: Sample results of a search for "hip replacement" using DualKey "HIPREP"

Count	Description Identifier	Concept Identifier	Term
1	1480791012	386649003	partial hip replacement by prosthesis
2	33592011	19954002	total replacement of hip with use of methyl methacrylate
3	50150016	29969002	replacement of acetabulum of hip
4	54398014	32581000	partial hip replacement by cup with acetabuloplasty
.....
11	183737015	112728000	total revision of hip replacement with use of methyl methacrylate

7.6.1.2 Multiple keywords



The performance of single *keyword* searches is highly dependent on the number of candidate *descriptions* returned by the *keyword* for subsequent filtering. The extremely high number of matches for some words in common use makes it likely that some searches will be unacceptably slow.

One way to alleviate this problem would be to create a table containing a row for all combinations of word pairs in each *description*. In some database environments that support optimization of multiple key searches, this may offer no benefits. However, in other environments, such a table may substantially speed searches.

A comprehensive word pair table would be very large. Such a table covering the full content of *SNOMED CT* would contain approximately 1.5 million unique word pairs and 6 million rows. Limiting the unique keys to the first three letter of each word reduces the table size to a more readily optimized set of keys. This requires the final part of the search to be conducted using text comparison (since the keys are incomplete).

7.6.1.2.1 Generating the DualKey index



Although *Dualkey* indexes are available as part of the Developer Toolkit , it is important to know how this table is generated. *SNOMED CT* users that generate *Extensions* should follow the method outlined below to generate new entries in the *Dualkey* index, based on the *descriptions* in the *Extension*.

For each *description*, parse the text of the *term*:

- To avoid inappropriate case mismatches, convert all characters to the same case;
- Extract words by breaking at spaces, punctuation marks, and brackets;

- For each word of three characters or more that is not in the list of *excluded words*, extract the first 3 characters, and arrange the word fragments in alphabetical *order*;
- Generate the dual keys for this *description* by concatenating each word fragment with those that come after it in the list;
- For each dual key, add a row to the word pair tables.

7.6.1.2.1.1 Example: Generation of keywords for a sample Description



Table 210: Sample Description

Description Identifier	Concept Identifier	Term
33592011	19954002	Total replacement of hip with use of methyl methacrylate

- To avoid inappropriate case mismatches, convert all characters to the same case.

"TOTAL REPLACEMENT OF HIP WITH USE OF METHYLE METHACRYLATE"

- Extract words by breaking at spaces, punctuation marks, and brackets.

1. TOTAL;
2. REPLACEMENT;
3. OF;
4. HIP;
5. WITH;
6. USE;
7. OF;
8. METHYLE;
9. METHACRYLATE.

- For each word of three characters or more, that is not in the list of *excluded words*, extract the first 3 characters, and arrange the word fragments in alphabetical *order*.

1. HIP;
2. MET;
3. REP;
4. TOT;
5. USE.

Note:

"OF" is less than 3 characters and is an *excluded word*, "WITH" is an *excluded word* and "MET" is duplicated, so we only include it once.

- Generate the dual keys for this *description* by concatenating each word fragment with those that come after it in the list;
- For each dual key, add rows to the word pair tables.

Table 211: DescDualKey

Dual key	Description Identifier
HIPMET	33592011

Dual key	Description Identifier
HIPREP	33592011
HIPTOT	33592011
HIPUSE	33592011
METREP	33592011
METTOT	33592011
METUSE	33592011
REPTOT	33592011
REPUSE	33592011
TOTUSE	33592011

Table 212: ConcDualKey

Dual key	Concept Identifier
HIPMET	19954002
HIPREP	19954002
HIPTOT	19954002
HIPUSE	19954002
METREP	19954002
METTOT	19954002
METUSE	19954002
REPTOT	19954002
REPUSE	19954002
TOTUSE	19954002

7.6.1.2.2 Searching for Descriptions using the DualKey index



A search on the dual key index can only be carried out if the user enters a search *string* that contains at least two word fragments both of which are three characters or more in length. If the search *string* does not meet this criterion, the single *keyword* search mechanism must be used.

- The user-typed search *string* is converted to consistent case;
- The *string* is parsed, breaking at spaces and punctuation characters;
- For each word of three characters or more, extract the first 3 characters, and arrange the word fragments in alphabetical *order*;
- Create a dual key by concatenating the first two 3 letter word fragments;
- Use this dual key to look up exact matches on the word pair index;
- *Descriptions* found by searching on the word pair index are screened, to see if they contain the complete words in the original search *string*

7.6.1.2.2.1 Example: Search using word pair index



User searches for "PYRO* 1 OXYGEN*".

- The *string* is parsed, breaking at spaces and punctuation characters.

 1. "PYRO*";
 2. 1;
 3. "OXYGEN*".

- For each word of three characters or more, extract the first 3 characters, and arrange the word fragments in alphabetical *order*.

 1. "OXY";
 2. "PYR".

- Create a dual key by concatenating the first two 3 letter word fragments.

OXYPYR

- Use this dual key to look up exact matches on the word pair index.

Table 213: Sample results of a search for "PYRO* 1 OXYGEN*"

Dual key	Description Identifier	Description
OXYPYR	1969019	2,5-Dihydroxy-pyridine oxygenase
OXYPYR	22565018	pyrogallol 1,2-oxygenase
OXYPRY	104951019	2,5-Dihydroxy-pyridine oxygenase

- *Descriptions* found by searching on the word pair index are screened, to see if they contain the complete words in the original search *string* :
 - *Description* 1969019 is eliminated since it does not contain the word "1";
 - *Description* 104951019 is eliminated, it does not contain the word "1" or any word beginning with the *string* "pyro".

7.6.1.3 Using word equivalents to enhance searches



In healthcare, there are many words with equivalent meanings. *Synonyms* provide alternative phrases referring to the *concept*. However, *synonyms* are not created automatically for every possible combination of words with an equivalent meaning. The success of simple searches using one or more *keywords* depends on the text of the available *descriptions*. Therefore searches will fail or will be incomplete where a different equivalent word is used in the search.

For example: "Kidney stone" and "Renal calculus" are synonymous *descriptions* in *SNOMED CT*. A search of *SNOMED CT* for the target phrase "kidney stone fragmentation" yields the result "Percutaneous nephrostomy with fragmentation of kidney stone," while a search for "Renal stone fragmentation" yields no results.

One way of addressing this problem is to maintain a table of *word equivalents*. A table of this type is a prerequisite for exhaustive *synonym* generation. An initial set of *word equivalents* is included in the *SNOMED CT* Developer Toolkit. Individual implementers will wish to add additional *word equivalents* to meet the requirements of their particular medical specialty or user needs. This table is an additional resource to assist searching and parsing of phrases. It need not be a comprehensive dictionary of words. Many searches can be completed without reference to this table so it need not contain every word or equivalent phrase used in *SNOMED CT*.

Several factors complicate the initial population and subsequent use of the *word equivalents table*:

- A phrase of two or more words may be equivalent to a single word.

Example:

"Endoscopic esophagus examination" is equivalent to "esophagoscopy"

- A word may have more than one meaning, and in this, only one meaning of a pair of words may be equivalent. Thus an apparent enhancement of a search may in practice lose some of the specificity of the intended search.

Example:

"Tap" and "aspiration" are equivalent in the context of *terms* such as "pleural tap", "pleural aspiration", but not in the context of a "patella tap", a physical "tap" on a bag or catheter, or the clinical disorder "neonatal aspiration syndrome".

- When searching using incomplete words and/or wildcards, use of *word equivalents* may impede effective searches by increasing the number of spurious potential matches. This either extends the processing required to filter the real matches from the potential matches or increases the length of the list of choices presented to the user.

A wise system developer will allow the user to customize their search options, enabling searches to be narrowed, or extended to meet the needs of varying circumstances.

7.6.1.3.1 Example: Using word equivalents table to extend a failed search



A system user enters the search *string* "Fragmentation of renal calculus;" the search returns no results. The search application that the user has been provided with has the option to extend the search by using the *word equivalents table*. The user selects this option and searches again using the same search *string*.

The *word equivalents table* contains the following relevant entries:

Table 214: Word Equivalents Table Example

WordBlockNumber	WordText	WordType
1021	KIDNEY	2 (word equivalent)

WordBlockNumber	WordText	WordType
1021	RENAL	2 (<i>word equivalent</i>)
4430	CALCULUS	2 (<i>word equivalent</i>)
4430	CALCULI	1 (<i>word form variant</i>)
4430	STONE	2 (<i>word equivalent</i>)
9870	RENAL STONE	4 (<i>equivalent phrase</i>)
9870	KIDNEY STONE	4 (<i>equivalent phrase</i>)
9870	KIDNEY CALCULUS	4 (<i>equivalent phrase</i>)
9870	RENAL CALCULUS	4 (<i>equivalent phrase</i>)
9870	NEPHROLITH	2 (<i>word equivalent</i>)

The table is used to make substitutions in the search *string* to produce all possible unique search variants:

"Fragmentation of renal calculus"

"Fragmentation of renal stone"

"Fragmentation of kidney stone"

"Fragmentation of kidney calculus"

"Fragmentation of Nephrolith"

"Fragmentation of renal calculus"

"Fragmentation of renal calculi"

"Fragmentation of kidney calculi"

These 8 search *strings* are used as the target phrase for *keyword* searches on the word pair index. Results from all 8 searches are combined, and duplicate *concepts* are eliminated, giving the final list of search results.

7.6.1.4 Rationalizing searches that return duplicate hits



In the previous sections of this guide, we have considered methods of ensuring that searches on a target phrase maximize the possibility of finding the *concept* that the system user requires. It is equally important to prevent the search results from containing excessive matches, since these will require filtering by the user, imposing an additional burden. Some strategies for limiting the number of search results displayed are discussed in the following sub-sections.

7.6.1.4.1 Avoiding multiple hits on the same concept



In many instances several *synonyms* associated with the same *concept* contain the same *keyword*. The designer of search software may consider filtering the output of search results so that only the first matching *description* for a *concept* is displayed.

👉 Example:

"Endoscopic examination of the stomach" and "endoscopy of the stomach" are *synonyms* of the same *concept*. A search for the target phrase "endo* stomach" would return the first phrase found during the

search. The second would be excluded, since it has the same *concept identifier* as an existing match for this search.

7.6.1.4.2 Constraining and extending search parameters



User configurable options may be one way of limiting search results. Three possible methods of limiting search results through user configurable options are suggested here:

- Limiting searches to exact matches unless wild cards are used. A search on a single word may produce many matches if it is assumed that the user is searching for any phrase that contains the target word. Forcing the use of wild cards for this kind of search can help avoid this problem.
- Make searches that include use of "word equivalents" a user configurable option that can be used to extend or constrain a search.
- Display search results a few at a time, with most frequently used *descriptions* listed first. This option will require the application to track the frequency of *term* selection so that search results can be sorted in this way.

7.6.2 Hierarchical Navigation



This section of the guide describes the *Terminology services* that are likely to be required to navigate *SNOMED CT* hierarchies.

One of the key strengths of *SNOMED CT* is a rich set of *relationships* that connect the *concepts* within the terminology. The primary use of these *relationships* is to facilitate selective retrieval. However, some of these *relationships* are arranged in hierarchies that can be navigated using an appropriate user-interface control. For example, the *subtype hierarchy* formed by the *|* is a *relationship* can be used to navigate from a selected *concept* to another *concept* that has a more specific or less specific meaning.

SNOMED CT also specifies standard ways to represent multiple *navigation* hierarchies that can be designed to meet different requirement. Unlike *relationship* based hierarchies, *navigation* hierarchies convey no semantic information but are intended to be used to enhance the user experience when navigating through the terminology.

7.6.2.1 Access to hierarchically related concepts



Terminology servers should enable client applications to access collections of *Concepts* that are related to a specified *Concept* as:

- *Subtype children*
- *Subtype descendants* (includes all generations of *children*);
- *Supertype parents*
- *Supertype ancestors* (includes all previous generations of *parents*).

7.6.2.2 Using *|* is a *|Relationships* for hierarchy navigation

7.6.2.2.1 The SNOMED CT hierarchy

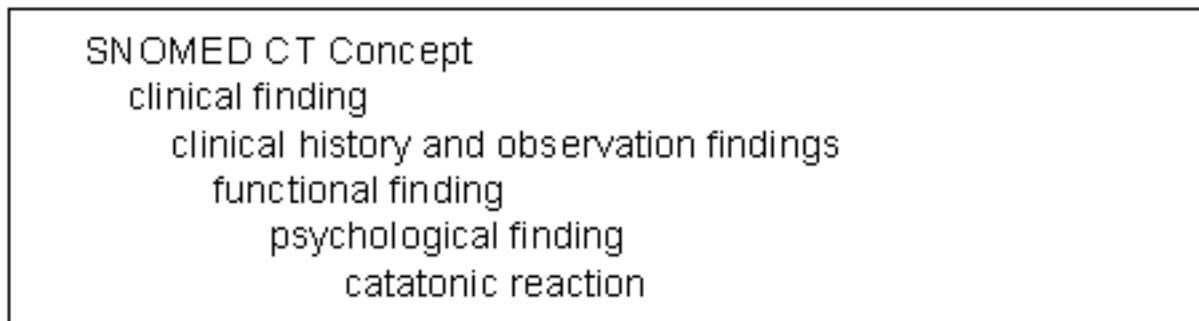


The "*SNOMED CT hierarchy*" refers to the organization of *concepts* in *SNOMED CT* from the general, at the top of the *hierarchy*, to the more specific or "granular" at the bottom. The *concepts* that make up the very top level of the *hierarchy* are shown in [Table 215](#). All other *SNOMED CT* *concepts* fall under one or more of these categories.

Table 215: Top Level Concepts

<ul style="list-style-type: none"> • Clinical finding • Procedure • Observable entity • Body structure • Organism • Substance • Pharmaceutical / biologic product • Specimen • Special concept • Linkage concept 	<ul style="list-style-type: none"> • Physical force • Event • Environment or geographical location • Social context • Situation with explicit context • Staging and scales • Physical object • Qualifier value • Record artifact
--	---

Several levels of increasingly fine categorization may exist between the top level of the *hierarchy* and *concepts* that have sufficient detail to be recorded in a patient's medical record. [Figure 102](#) shows the levels of *hierarchy* that exist between the top-level *Concept* | Clinical finding | and the finding "Catatonic reaction."

**Figure 102: Hierarchy example: Catatonic Reaction**

7.6.2.2.2 Hierarchy Representation in the relationships table



The *SNOMED CT Relationship* table represents *relationships* between one *SNOMED CT concept* and another by including a row in the table for each such *relationship*. The columns *sourcelId*, *typelId* and *destinationId* define the source of the *relationship*, the kind of *relationship* that exists and the target of the *relationship* respectively. Each of these fields, contains a *SNOMED CT Concept Identifier*. Hierarchical *relationships* are expressed by linking the source *concept* to its "parents" (i.e. the *concept* or *concepts* immediately above it in the *hierarchy*). The *typelId* used to represent the *subtype hierarchy* is the | is a | *relationship*.

For example, we can say | catatonic reaction | | is a | | psychological finding |. This is expressed in the *Relationship Table* as follows:

Table 216: Subtype Relationship Example

<i>sourcelId</i>	<i>typelId</i>	<i>destinationId</i>
102909009	116680003	116367006

Where:

- 102909009 is the *concept identifier* for | catatonic reaction |;
- 116680003 is the *concept identifier* for the | is a | *relationship*;
- 116367006 is the *concept identifier* for | psychological finding |.

Conversely, by inverting the **| is a** relationship we can find the *children* of the target *Concept*, (i.e. the *Concept* or *Concepts* immediately below it in the *hierarchy*).

7.6.2.2.3 Using | is a |Relationships to enhance search capabilities



This section is concerned with the ways in which the *hierarchy* can be used to help a *SNOMED CT* user when they are searching or browsing the terminology.

 **Note:** The primary use of the *SNOMED CT subtype hierarchy* is to support effective retrieval and aggregation of data. This is discussed in [Testing and traversing subtype relationships](#).

It is possible to start at the top of *hierarchy* and navigate from parent to *child* in order to find a *Concept* or *term* in *SNOMED CT*. A more efficient approach, however, is to use the *hierarchy* to supplement a *keyword* search by enabling the user to look at related *Concepts* in order to consider them as alternative matches, or to check the context of a search result. The following examples illustrate these two uses of the *SNOMED CT hierarchy*.

 **Example:**

1. Checking supertypes:

- A user wishes to find a *description* that relates to the condition of a patient who is hypersensitive to an allergen. The user performs a search on the *keyword* "Hypersensitivity" and finds an exact match. Before the user selects the *description* for inclusion in the patient record, they check the *Fully Specified Name*, which is "Sensitivity (finding)." The user then checks the *hierarchy* and discovers that the selected *Concept* has "Psychological finding" as an *ancestor*, which indicates that this is not the correct *description* to use in this context.

2. Checking subtypes:

- A user wishes to find a *description* that relates to the condition of a patient who is hypersensitive to an allergen. The user searches for the *keyword* "allergy," and finds one *Concept* having a *description* that is an exact match. The user then looks at the *children* of the *Concept* (i.e. those *concepts* immediately below it in the *hierarchy*). One of the *children* has the preferred *description* "Contact Hypersensitivity" which matches the user's intended meaning. The user selects this *Concept* for inclusion in the patient record.

7.6.2.2.4 Using | is a |Relationships to display hierarchical information in applications



Most visual application development tools contain a *component* designed to display hierarchical information as a tree in which branches can be expanded or collapsed. Tree views are well-suited to displaying *SNOMED CT* hierarchical Relationships (see [Figure 103](#)). These views are used in many different user-interfaces where information needs to be represented as a hierarchy (e.g. displaying a file-system as a hierarchy of folders or providing a collapsible outline of a document or help file). Therefore, most users will already be familiar this paradigm.

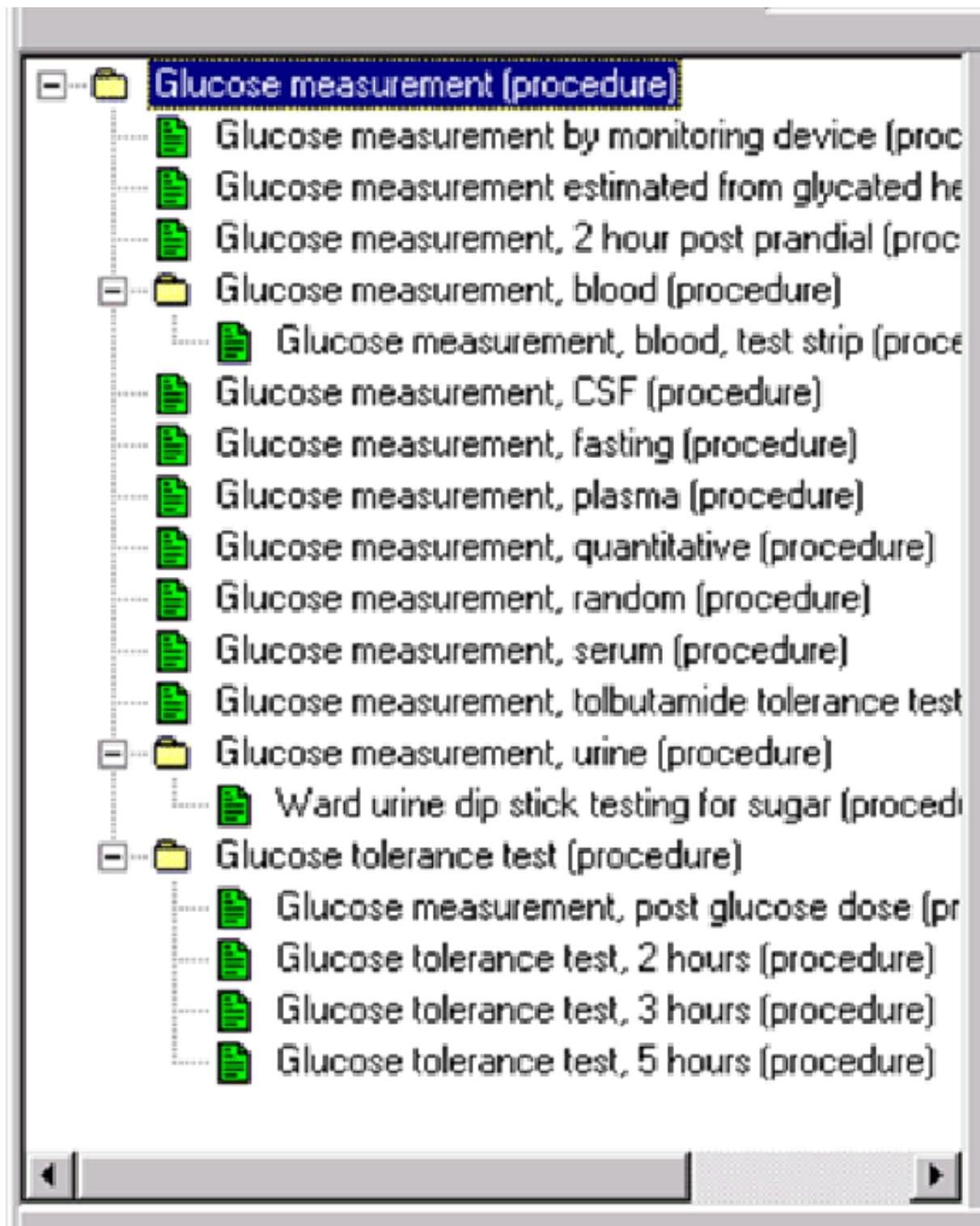


Figure 103: SNOMED CT hierarchy represented in a tree view

The process of creating a tree view from the *SNOMED CT Relationship* table is straightforward as long as a few simple ideas are mastered:

- Most standard tree-views controls start from a single root and require that higher level branches must be added before sub-branches. This means that when viewing part of the *hierarchy* from the bottom up, the tree must be compiled in temporary form before it can be displayed.

- Since the depth of the *hierarchy* is not known in any particular case, operations that iterate up or down the depth of the *hierarchy* must be done using a recursive algorithm. However, this recursion must usually be limited since placing the entirety of the *SNOMED CT* hierarchy in a single tree control is likely to create performance issues and may exceed physical limits on the capacity of the control.
- Standard tree view controls are not good at displaying the multiple parent nodes that occur in a *polyhierarchy* like *SNOMED CT*. Therefore, some compromises need to be made to present options for navigation up the hierarchy.
- Effective use of some tree controls requires unique keys for each node. Multiple parents and multiple roots through the hierarchy mean that the same *Concepts* will appear in multiple places in the hierarchy. Therefore, the *concept identifier* cannot be used to provide a key that is globally unique within the hierarchy.

7.6.2.3 Using | Part of |Relationships for hierarchy navigation



In addition to the *subtype hierarchy* represented by | is a | *relationships*, *SNOMED CT* also represents a partonomy hierarchy using | Part of | *relationship*. This creates an alternative hierarchy which can be also be used for navigation. The difference between these hierarchies is that:

- The *subtype hierarchy* relates *concepts* to supertypes that represent more general *concepts*. Each body structure *concept* has an | is a | *relationship* to one or more *concepts* that represents the *whole or any part of* the organ or other body part that contains it. *Concepts* that represent the *whole or any part of* an organ or body part are distinguished by their *fully specified names* which include the word 'structure'. These contrast with *concepts* that represent the *entirety* of an organ or body part which contain the word 'entire'.

 **Example:** | Right ventricular structure | | is a | | heart structure |

- The partonomy hierarchy relates body structure to *concepts* to *concept* that represent | the *entirety of* | or an organ or anatomical structure of which they form part

 **Example:** | Entire right ventricle | (is) | part of | | entire heart |

 **Note:** In everyday speech the word "heart" may mean either | heart structure | or | entire heart | and the distinction between them is often overlooked. However, from a semantic perspective the difference is highly significant. The removal of some part of an organ does not imply the removal of the entire organ. Thus, while it is correct to state that | Right ventricular structure | | is a | | heart structure |, it would be wrong to state that | Entire right ventricle | | is a | | entire heart | or | Right ventricular structure | | is a | | entire heart |.

7.6.2.4 Using other Relationships to navigate SNOMED CT content



Many *SNOMED CT Concepts* have *relationships* with content in other areas of terminology. These *Relationships* are one of the ways in which *SNOMED CT* provides computer readable definitions for medical *concepts*. For example, diseases in *SNOMED CT* generally have a *Relationship* to the body site affected by the disorder and a *Relationship* to the morphology associated with the disease. Procedures in *SNOMED CT* might have *Relationships* to the *concept*, which defines the type of surgical action being carried and the procedure site, for example. Examples of *Relationships* for a disease and a procedure are shown below. A full list of the *Relationships* that can be used for each type of *Concept* can be found in [Table 3](#).

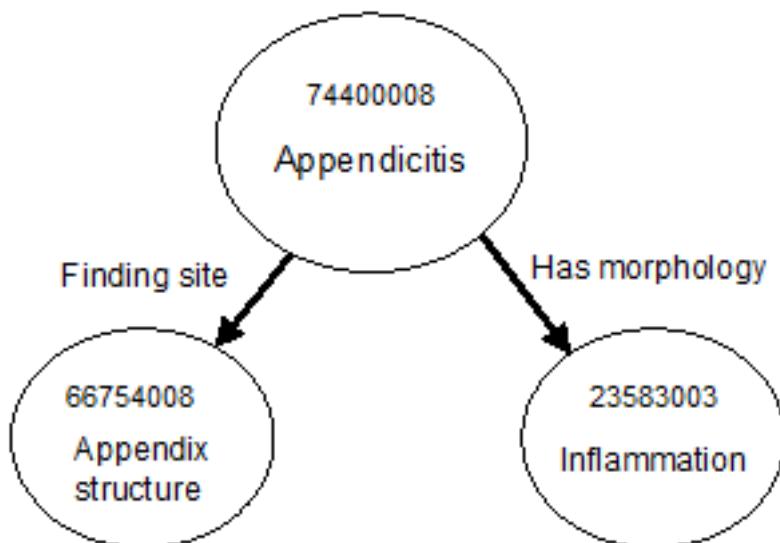


Figure 104: Relationship for disease appendicitis

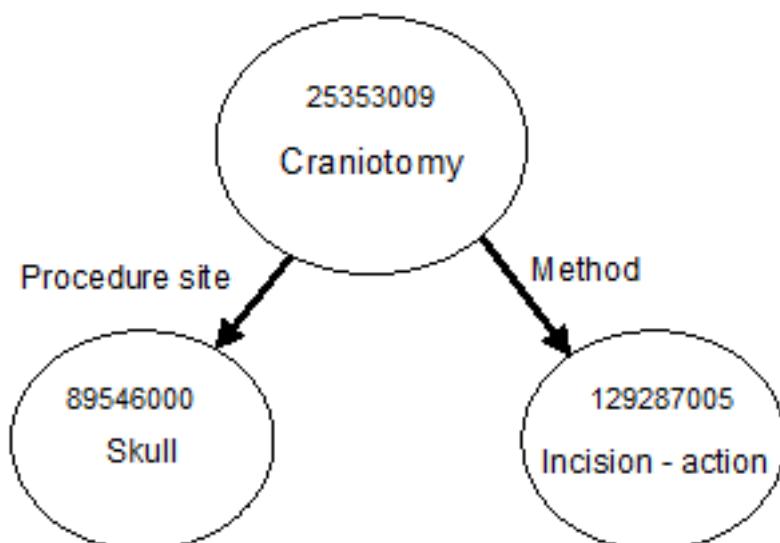


Figure 105: Relationships for procedure craniotomy

These *Relationships* are very useful in the context of data retrieval and analysis. The *Relationships* can also be used to aid in the search for specific *SNOMED CT Concepts* in cases where the *term* alone may not sufficiently distinguish between choices. For example, a search for all inflammatory diseases of the lung could be carried out as follows:

- Use the *hierarchy* to compile a list of all *Concepts* that are lung structures;
- Search for any *Concept* that has a row in the *Relationships table* with a *Relationship Type* of "Disorder site," and with *ConceptId2* included in the list of lung structures;
- Now exclude any procedures from the list that do have the | Associated morphology | "Inflammation" in the *Relationship* table;
- Final product | is a | list of all lung disorders that involve inflammation.

To achieve these same results with a string search we would have to perform separate searches for | pneumonia |, bronchitis, | pleurisy | and many other conditions that cannot be linked via a simple string search.

7.6.2.5 Implementing Navigation Hierarchies



This section demonstrates how an *Ordered Reference Set* is used to specify and display a customized *navigation hierarchy*. A *navigation hierarchy* is a hierarchical view of SNOMED CT concepts which may differ from the strict *subtype hierarchy* (represented by | is a | relationships).

7.6.2.5.1 Navigation Hierarchy Example



To illustrate the way a *navigation hierarchy* is represented this section uses an example containing a set of *concepts* used to describe x-ray examinations of the upper and lower limbs. The resulting *navigation hierarchy* might usefully be extended to include other x-ray procedures but has been kept small for the purposes of the example.

Reference sets are created as explained in [how to create a new Reference Set using an existing pattern](#) and their *Identifier*, name and type are specified by a *concept*. The example *reference set* would be specified by a *concept* with the characteristics shown in [Table 217](#).

Table 217: Concept specifying the Example Navigation Reference Set

Characteristic	Value	Comment
id	<RefsetId-A>	These symbolic values are used to avoid any potential confusion with released <i>reference sets</i> . The <i>moduleId</i> represents the module in which the <i>reference set</i> was developed.
moduleId	<ModuleId-A>	
preferredTerm	Example Navigation Reference Set	
is a	Ordered type reference set	

The *concepts* included in the *reference set* are shown with their *preferred terms* in [Table 218](#).

Table 218: Concepts used in the Example Navigation Reference Set

Id	Preferred Term
1225002	radiography of humerus
1597004	skeletal X-ray of ankle and foot
168594001	clavicle X-ray
168619004	plain X-ray head of humerus
168620005	plain X-ray shaft of humerus
168623007	X-ray shaft of radius/ulna
168637003	plain X-ray radius
168655007	instability views carpus

Id	Preferred Term
168663008	plain X-ray head of femur
168664002	femoral neck X-ray
168665001	plain X-ray shaft of femur
168669007	patella X-ray
205115004	radiologic examination of femur, anteroposterior and lateral views
241063007	bicipital groove X-ray
241066004	ulna groove X-ray
241069006	ulna X-ray
241071006	scaphoid X-ray
241073009	metacarpal X-ray
241075002	femur X-ray
241076001	tibia and/or fibula X-ray
241077005	tibia X-ray
241078000	fibula X-ray
241079008	metatarsal X-ray
241080006	tarsus X-ray
268427003	X-ray shaft of tibia/fibula
271311001	carpal bones X-ray
302402006	radius and/or ulna X-ray
37815002	diagnostic radiography of calcaneus
40348008	skeletal X-ray of pelvis and hip
418687005	fluoroscopy of humerus

Id	Preferred Term
427961005	x-ray of acetabulum
432552002	computed tomography of clavicle
48966008	skeletal X-ray of shoulder and upper limb
5433008	skeletal X-ray of lower limb
70780000	skeletal X-ray of elbow and forearm
72872009	skeletal X-ray of upper limb
79082005	diagnostic radiography of fibula, combined AP and lateral
82420003	radiologic examination of forearm, anteroposterior and lateral views

The members of the example *reference set* would be distributed in a file with a name like:

- xder2_ciRefset_NavigationRefsetExampleFull_XX_20100731.txt.

The content of this file is shown in [Table 219](#) and the resulting hierarchical display is shown in [Figure 106](#).

Table 219: Example Navigation Reference Set File

id	effectiveTime	active	moduleId	refsetId	referencedComponentId	order	linkedId
<uuid-0211>	20100731	1	<ModuleId-A>	<RefsetId-A>	<RefsetId-A>	1	5433008
<uuid-0212>	20100731	1	<ModuleId-A>	<RefsetId-A>	5433008	1	241080006
<uuid-0213>	20100731	1	<ModuleId-A>	<RefsetId-A>	241080006	1	37815002
<uuid-0214>	20100731	1	<ModuleId-A>	<RefsetId-A>	5433008	2	241079008
<uuid-0215>	20100731	1	<ModuleId-A>	<RefsetId-A>	5433008	3	241076001
<uuid-0216>	20100731	1	<ModuleId-A>	<RefsetId-A>	241076001	1	241078000
<uuid-0217>	20100731	1	<ModuleId-A>	<RefsetId-A>	241078000	1	268427003
<uuid-0218>	20100731	1	<ModuleId-A>	<RefsetId-A>	241078000	2	79082005
<uuid-0219>	20100731	1	<ModuleId-A>	<RefsetId-A>	241078000	3	241077005
<uuid-0220>	20100731	1	<ModuleId-A>	<RefsetId-A>	5433008	4	241075002

id	effectiveTime	active	moduleId	refsetId	referencedComponentId	order	linkedId
<uuid-0221>	20100731	1	<ModuleId-A>	<RefsetId-A>	241075002	1	205115004
<uuid-0222>	20100731	1	<ModuleId-A>	<RefsetId-A>	241075002	2	168665001
<uuid-0223>	20100731	1	<ModuleId-A>	<RefsetId-A>	241075002	3	168664002
<uuid-0224>	20100731	1	<ModuleId-A>	<RefsetId-A>	241075002	4	168663008
<uuid-0225>	20100731	1	<ModuleId-A>	<RefsetId-A>	5433008	5	168669007
<uuid-0226>	20100731	1	<ModuleId-A>	<RefsetId-A>	5433008	6	1597004
<uuid-0227>	20100731	1	<ModuleId-A>	<RefsetId-A>	5433008	7	40348008
<uuid-0228>	20100731	1	<ModuleId-A>	<RefsetId-A>	40348008	1	427961005
<uuid-0229>	20100731	1	<ModuleId-A>	<RefsetId-A>	<RefsetId-A>	2	72872009
<uuid-0230>	20100731	1	<ModuleId-A>	<RefsetId-A>	72872009	1	302402006
<uuid-0231>	20100731	1	<ModuleId-A>	<RefsetId-A>	302402006	1	241069006
<uuid-0232>	20100731	1	<ModuleId-A>	<RefsetId-A>	241069006	1	168623007
<uuid-0233>	20100731	1	<ModuleId-A>	<RefsetId-A>	302402006	2	168637003
<uuid-0234>	20100731	1	<ModuleId-A>	<RefsetId-A>	302402006	3	70780000
<uuid-0235>	20100731	1	<ModuleId-A>	<RefsetId-A>	70780000	1	241066004
<uuid-0236>	20100731	1	<ModuleId-A>	<RefsetId-A>	302402006	4	82420003
<uuid-0237>	20100731	1	<ModuleId-A>	<RefsetId-A>	72872009	2	168594001
<uuid-0238>	20100731	1	<ModuleId-A>	<RefsetId-A>	168594001	1	432552002
<uuid-0239>	20100731	1	<ModuleId-A>	<RefsetId-A>	72872009	3	1225002
<uuid-0240>	20100731	1	<ModuleId-A>	<RefsetId-A>	1225002	1	241063007
<uuid-0241>	20100731	1	<ModuleId-A>	<RefsetId-A>	1225002	2	168620005
<uuid-0242>	20100731	1	<ModuleId-A>	<RefsetId-A>	1225002	3	168619004
<uuid-0243>	20100731	1	<ModuleId-A>	<RefsetId-A>	1225002	4	418687005

id	effectiveTime	active	moduleId	refsetId	referencedComponentId	order	linkedId
<uuid-0244>	20100731	1	<ModuleId-A>	<RefsetId-A>	72872009	4	168655007
<uuid-0245>	20100731	1	<ModuleId-A>	<RefsetId-A>	72872009	5	271311001
<uuid-0246>	20100731	1	<ModuleId-A>	<RefsetId-A>	271311001	1	241071006
<uuid-0247>	20100731	1	<ModuleId-A>	<RefsetId-A>	72872009	6	241073009
<uuid-0248>	20100731	1	<ModuleId-A>	<RefsetId-A>	72872009	7	48966008

👉 **Note:** Each of the symbolic names '<uuid-0211>' to '<uuid-0248>' in [Table 219](#) represents a unique 128-bit *UUID* generated by a standard algorithm. Use of the same symbolic names elsewhere in this example indicates a revised version of the same component with the same *Identifier*. Use of the same symbolic name in another example does not imply the same *Identifier*.

- 5433008 | skeletal X-ray of lower limb |
 - 241080006 | tarsus X-ray |
 - 37815002 | diagnostic radiography of calcaneus |
 - 241079008 | metatarsal X-ray |
 - 241076001 | tibia and/or fibula X-ray |
 - 241078000 | fibula X-ray |
 - 268427003 | X-ray shaft of tibia/fibula |
 - 79082005 | diagnostic radiography of fibula, combined AP and lateral |
 - 241077005 | tibia X-ray |
 - 241075002 | femur X-ray |
 - 205115004 | radiologic examination of femur, anteroposterior and lateral views |
 - 168665001 | plain X-ray shaft of femur |
 - 168664002 | femoral neck X-ray |
 - 168663008 | plain X-ray head of femur |
 - 168669007 | patella X-ray |
 - 1597004 | skeletal X-ray of ankle and foot |
 - 40348008 | skeletal X-ray of pelvis and hip |
 - 427961005 | x-ray of acetabulum |
- 72872009 | skeletal X-ray of upper limb |
 - 302402006 | radius and/or ulna X-ray |
 - 241069006 | ulna X-ray |
 - 168623007 | X-ray shaft of radius/ulna |
 - 168637003 | plain X-ray radius |
 - 70780000 | skeletal X-ray of elbow and forearm |

- 241066004 | ulna groove X-ray |
- 82420003 | radiologic examination of forearm, anteroposterior and lateral views |
- 168594001 | clavicle X-ray |
 - 432552002 | computed tomography of clavicle |
- 1225002 | radiography of humerus |
 - 241063007 | bicipital groove X-ray |
 - 168620005 | plain X-ray shaft of humerus |
 - 168619004 | plain X-ray head of humerus |
 - 418687005 | fluoroscopy of humerus |
- 168655007 | instability views carpus |
- 271311001 | carpal bones X-ray |
 - 241071006 | scaphoid X-ray |
- 241073009 | metacarpal X-ray |
- 48966008 | skeletal X-ray of shoulder and upper limb |

Figure 106: Example Navigation Reference Set - Hierarchy View

This *reference set* could be updated by addition of the rows in a subsequent release. If the three rows shown in [Table 220](#) are added in the next version, the results are as follows:

- 48966008 | skeletal X-ray of shoulder and upper limb | is removed from the *reference set* because the row with id=<uuid-0248> and the most recent *effectiveTime* is now inactive (active=0);
- The order of 241073009 | metacarpal X-ray | and 271311001 | carpal bones X-ray | are reversed as the most recent row for id=<uuid-0245> has order=6 while the most recent row for id=<uuid-0247> has order=5.

The changed part of the hierarchy is shown in [Figure 107](#).

Table 220: Example Navigation Reference Set File - Updated Rows

id	effectiveTime	active	moduleId	refsetId	referencedComponentId	order	linkedId
<uuid-0245>	20110731	1	<ModuleId-A>	<RefsetId-A>	72872009	6	271311001
<uuid-0247>	20110131	1	<ModuleId-A>	<RefsetId-A>	72872009	5	241073009
<uuid-0248>	20110131	0	<ModuleId-A>	<RefsetId-A>	72872009	7	48966008

- 5433008 | skeletal X-ray of lower limb |
 - ... *unchanged* ...
- 72872009 | skeletal X-ray of upper limb |
 - ... *unchanged* ...
 - 168655007 | instability views carpus |
 - 241073009 | metacarpal X-ray |
 - 271311001 | carpal bones X-ray |

- 241071006 | scaphoid X-ray |

Figure 107: Example Navigation Reference Set - Updated Hierarchy View

7.6.2.5.2 Navigation Hierarchy Inheritance



A Navigation Reference Set may organize some *concepts* while allowing the *subtype hierarchy* (or another *navigation hierarchy*) to provide additional hierarchical links. In this case, a *concept* that has no children in the *navigation hierarchy* inherits the children specified in the *subtype hierarchy* or a specified default *navigation hierarchy*.

7.6.2.6 Using Tree View Components for Hierarchy Display



The two examples given below show the creation of a tree view from a small sample *hierarchy*.

The principals used can be extended to any size or depth of *hierarchy*.

7.6.2.6.1 Example 1: Show all descendants of Concept "A" in a tree view



Table 221: Example Relationships

ConceptId1	Relationship	ConceptId2
B	is a	A
C	is a	A
D	is a	B
E	is a	B
E	is a	C
C	is a	F

We must process each *concept* in the *hierarchy*, starting at 'A'. Add a tree node for 'A', and then *query* to get the *children* of 'A'. Process each *child* recursively, i.e. add a node to the tree view for the *child*, then *query* for its *children*, etc.

Table 222: Child nodes

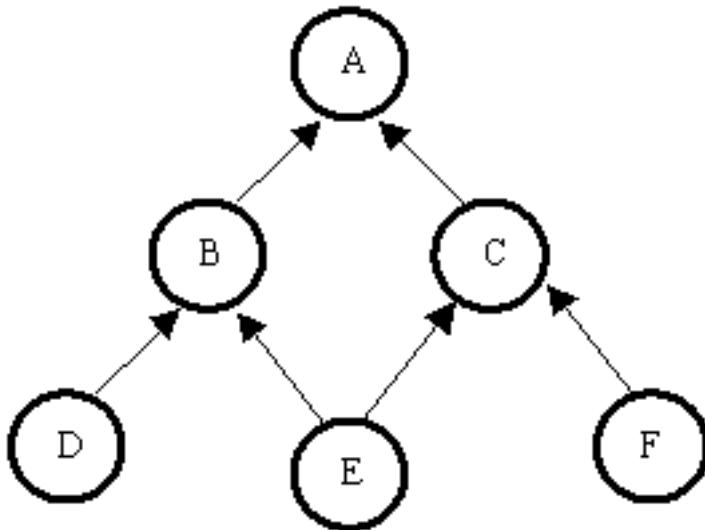
Node	Child Node
1	2
1	5
2	3
2	4
5	4

Node	Child Node
5	6

Table 223: Concept to node cross reference

Node	Concept Identifier
1	A
2	B
3	D
4	E
5	C
6	E
7	F

Now we have tree nodes and their *children* for each *Concept*. If the nodes have been added to a Windows tree view component, display will be automatic. If a text-based display is being used then the nodes can be output to the screen using the indent style display. Note that the *Concept 'E'* appears in the tree view twice, under each of its parents.

**Figure 108: Tree view of sample hierarchy - descendants of "A"**

7.6.2.6.2 Example 2 - Show all ancestors of Concept "E"



In order to construct the tree view, we must start from the top down, so we must create a temporary view of the *hierarchy* before we can add nodes to the tree view. Query to get the parents of 'E'. Process each parent recursively, i.e. add an entry to the temporary table, stating that 'E' | is a | child of each

of its parents, then *query* to get its parent, etc. When the top of the tree is reached, a record is kept of the top-level *concept*, since this will be the starting point for building the tree view.

Table 224: Temporary view of the hierarchy

Concept	Child Concept
B	E
C	E
A	B
A	C

We can now use the temporary table information to build the tree view from the top down. Starting at A, add a node to the tree view. Work recursively from the information in the temporary view of the *hierarchy* to add the *descendants* of 'A' into the tree view.

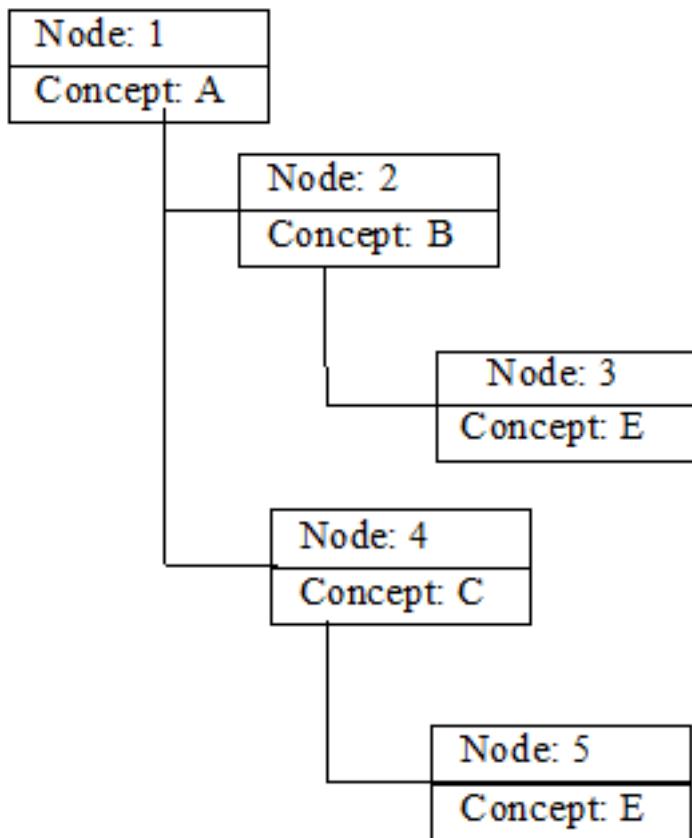


Figure 109: Tree view of sample hierarchy - ancestors of "E"

7.6.3 Applying Reference Sets



Refsets can be used for many different purposes. This section outlines some of the ways in which the Refset mechanism specified in this guide can be used to meet different practical requirements. The uses outlined are illustrative examples and do not represent all possible applications of Refsets.

7.6.3.1 Languages and Dialects



The *Language Reference Set* applicable to the local language, *dialect* and term preferences should be used to filter or prioritize the display of matching terms.

A *SNOMED CT enabled application* should be able to:

- Allow a selection of a particular language as a configuration option;
- Restrict access to *descriptions* so that are acceptable or preferred in the selected language;
- Treat the *synonym* marked as preferred in the selected language as the default *preferred term*;
- Treat the *Fully Specified Name* marked as preferred in the selected language as the *Fully Specified Name* to be displayed where relevant.

An application may support multiple *languages* or *dialects* allowing selection of combinations of Language Reference Sets. In this case, the primary selected language is supplemented by acceptable *Descriptions* in supplementary languages or *dialects*.

7.6.3.2 National requirements for specific Concepts



SNOMED CT is designed for use in many different countries and consequently includes a certain number of country-specific *Concepts*. Each country, may have specific requirements for the representation of *Concepts* that are not meaningful in other countries. These variations are particularly significant for the interfaces between clinical care, service administration and reimbursement. National laws and conventions may also create additional *refinements* of more general *Concepts*.

Reference Sets included in a National *Extension* release can be used to configure searches performed by *terminology services* to meet National requirements.

- *Simple Reference Sets* can be used to filter search contents;
- *Ordered Reference Sets* can be used to prioritize search result or provide a natural ordering of search results;
- *Attribute Value Reference Sets* can be used to filter information based on particular criteria represented by the *valueId* of particular members of the *Reference Set*.
- *Annotation Reference Sets* can be used to supplement the search results with information annotated to a *component* providing advice on intended use.

A *SNOMED CT enabled application* should be able to apply National *Reference Sets* to filter searches.

7.6.3.3 Regional variations in disease prevalence



There are substantial differences in the prevalence of diseases in different regions in which *SNOMED CT* may be used. Users will expect to find the conditions they commonly deal with, without being distracted by long lists of conditions they rarely see.

A *SNOMED CT enabled application* should be able to select *Reference Sets* that represent prevalence characteristics for a particular region. Based on the selected configuration searches should:

- Selectively include or exclude *concepts* or *descriptions* based on presence in or absence from a selected *Simple Reference Set*;
- Prioritize access to *concepts* or *descriptions* based on the order specified in a selected *Ordered Reference Set*.

The way in which access is prioritized depends on the nature of the application and its operating environment. However, examples of prioritization include:

- Showing *descriptions* associated with high priority *concepts* before those with lower priority when searching for word or phrases;
- Showing *concepts* with high priority before their less highly prioritized siblings in hierarchical displays;
- Initially listing *concepts* and associated *descriptions* with priority above a specified threshold and requiring an additional step to access those assigned lower priority.

7.6.3.4 Specialty and discipline-dependent variations in use of Concepts



SNOMED CT contains *Concepts* used by many different groups of health professionals. The frequency of use of these *Concepts* depends on the professional discipline and/or clinical specialty of the user. It is important to ensure that the user is able to access the *Concepts* that they use frequently, without being distracted by thousands of textually similar *Concepts* they rarely require.

A *SNOMED CT enabled application* should be able to select *Reference Sets* that represent the requirements of a particular speciality. Based on the selected configuration searches should:

- Selectively include or exclude *concepts* or *descriptions* based on presence in or absence from a selected *Simple Reference Set*;
- Prioritize access to *concepts* or *descriptions* based on the order specified in a selected *Ordered Reference Set*.

7.6.3.5 Local needs of organizations or individual users



The previous sections have dealt with requirements of countries, regions and specialties. Organizations and individual users may also have similar requirements for restricting or prioritizing access to particular *Concepts*.

A *SNOMED CT enabled application* should be able to rationally combine *Reference Sets* that represent National, Regional, specialty and local requirements. Based on the selected configuration searches should:

- Selectively include or exclude *concepts* or *descriptions* based on presence in or absence from selected *Simple Reference Sets*;
- Prioritize access to *concepts* or *descriptions* based on the order specified in the selected *Ordered Reference Sets*.

7.6.3.6 Supporting data entry protocols



Many clinical applications include facilities for data entry to be controlled or assisted by protocols, templates or structured data entry forms. Different sets of candidate *terms* or *concepts* may be appropriate to each data entry field. The sets of candidate *terms* or *concepts* for a field may be very large (e.g. any operative procedure) or very small (e.g. the possible observations from a particular examination).

Reference sets can be used to restrict the available options to match the requirements of a particular data entry protocol. *Reference Sets* provided by the author of the protocol can be applied to particular fields on a screen or particular data entry steps to configure relevant searches.

- *Simple Reference Sets* can be used to filter search contents;
- *Ordered Reference Sets* can be used to prioritize search result or provide a natural ordering of search results;
- *Attribute Value Reference Sets* can be used to filter information based on particular criteria represented by the *valueld* of particular members of the *Reference Set*.
- *Annotation Reference Sets* can be used to supplement the search results with information annotated to a *component* providing advice on intended use.

A *SNOMED CT enabled application* should be able to dynamically select a particular configuration based on identification of the data entry step or context. It should then be able to apply the relevant *Reference Sets* to provide an appropriate set of data entry options and/or to constrain text searches.

7.6.3.7 Managing the coded content of messages



A *Simple Reference Set* may be used to represent *value set* applicable to a particular field in a message. The entry of data to populate that field in the message can be constrained by filtering searches so that only *concept* in that *Reference Set* are returned.

Healthcare messages include fields that can be populated with codes from clinical coding schemes. *SNOMED CT* provides *concept identifiers* as a means of encoding *Concepts*. These *concept identifiers* are suitable for use in appropriate fields of many clinical messages.

Implementations of clinical messaging typically constrain the range of values that can be applied to particular fields. There are several *reasons* for this:

- To ensure that the information encoded is meaningful as a value for the specified field.

Example:

A field that is intended to describe the nature of investigation may contain a code that means "Serum glucose measurement" but should not contain a code that means "Hypoglycemia."

- To ensure that receiving application is able to process the message.

Example:

A locally added code value may be valid in a particular application but should not be used if the receiving application needs to retrieve, process or analyze the coded part of the message.

- To ensure adequate detail and specificity.

Example:

A field used to report an operative procedure could contain a code for "Abdominal procedure." However, this would not be adequate to meet the business purpose served by a message.

- To avoid unnecessary detail or diversity.

Example:

A biochemical investigation could be reported using a code that represents various detailed aspects of the method used to perform the investigation. Such details may be unnecessary to a clinician and may complicate the analysis, charting and graphing of a series of results reported at different levels of detail.

7.6.4 Access to qualifiers and refinable characteristics



A *terminology server* should enable an application to review the refinable *defining characteristics* and the specified set of *qualifying characteristics* for any selected *Concept*.

7.7 Testing and traversing subtype Relationships



The *subtype hierarchy* represented by | is a | relationships is an essential element in the structure and semantics of *SNOMED CT*. All *SNOMED CT enabled terminology servers* need to provide functions that test and traverse these *relationships* to navigate the hierarchy and to determine whether a *concept* is a *subtype* of another specified *concept*.

7.7.1 Top-level ancestor checking



Terminology servers should allow client applications to rapidly determine the top-level *Concept* that is the *supertype ancestor* for any specified *Concept*.

Each *Concept* has only one top-level supertype and this represents the semantic-type of the *Concept*.

7.7.2 Navigation concept checking



Terminology servers allow client applications to determine whether a specified *Concept* | is a | *navigation Concept*.

7.7.3 Subtype descendant testing



Terminology servers should be able to test whether any specified *Concept* | is a | *descendant subtype* of another specified *Concept*.

7.7.4 Subtype search scope restriction



Terminology servers should be able to restrict searches so that they return only those *Concepts* and (or their associated *Descriptions*) that are *subtype descendants* of a specified *Concept*.

Subtype search scope restriction is particularly valuable with respect to top-level *Concepts*. For example, when searching for a procedure it is useful to be able to exclude disorders or findings that may contain similar words or phrases.

Generalizing *subtype* search scope restriction to other nodes in the *subtype hierarchy* may significantly enhance usability in some situations.

👉 Example:

When undertaking an ophthalmologic examination, a search for findings could be constrained to findings related to the eye, increasing the specificity of results of searches for phrases containing the word "fundus."

7.7.5 Optimizing concept subsumption testing



Rapid and efficient computation of whether a *concept* | is a | *subtype descendant* of another *concept* is essential for effective *transformation of expressions* and for testing subsumption between *expressions*.

7.7.5.1 Approaches to concept subsumption testing



The *SNOMED CT* Technical Implementation Guide discusses several strategies for delivering efficient computation of subsumption between *concepts*. These are briefly summarized here with a brief evaluation of their suitability.

7.7.5.1.1 Recursive testing of subtype Relationships



It is possible to determine whether one *concept* subsumes another *concept* by recursively following every possible sequences of | is a | *Relationships* from a candidate *concept* until the predicate *concept* is reached or until all possible paths have been exhausted.

This approach is far too slow to deliver effective implementations in all environments in which it has been tested to date.

7.7.5.1.2 Semantic type Identifiers and hierarchy flags



Flags added to the internal representation of each *Concept* can be used to indicate the set of high-level *concept* nodes of which that *concept* is a *subtype*. A *concept* can only subsume *concepts* that include the same set of high-level *concept* flags. This approach can reduce the number of tests that need to be performed to recursively test the *subtype relationships*:

- If a candidate does not have all the high-level node flags that the predicate has, no further tests are needed. The candidate is not a *subtype* of the predicate.
- Even if a candidate shares the high-level node flags with the predicate, any path that reaches a *concept* that does not share those flags need not be further tested.

While faster than the unaided recursive testing approach, this is too slow to deliver effective implementations and is not scalable.

7.7.5.1.3 Use of proprietary database features



Some databases include additional features to support the recursive testing of a chain of hierarchical *relationships*. Other methods of optimization that may be applied to allow more rapid computation of *subtype descendant relationships* are outlined in the following subsections.

Current experiences of databases that support this type of approach indicate that (while easy to implement) the performance is substantially inferior to use of branch-numbering or transitive closure.

7.7.5.1.4 Branch numbering



The internal representation of each *Concept* can be extended to include a branch-number and a set of branch-number -ranges.

A branch-numbering algorithm can then be applied when each release of *SNOMED CT* is imported.

A typical branch-numbering algorithm processes the *subtype hierarchy* in the following way:

- A depth first tree walk is performed starting from the root *Concept* (branch-number 1) and an incrementing number is applied to each *Concept* when it is encountered for the first time.
- After the branch numbers have been computed a further tree walk allocates one or more branch-number ranges to each *Concept* with any *subtype descendants*:
 - Many *Concepts* will have a single branch number range containing all their *descendants*.
 - Some *Concepts* will have several non-contiguous ranges of *descendant Concept* branch numbers:
 - This is because a *Concept* may have multiple supertypes. Therefore, the *descendants* of a *Concept* may have branch numbers that were allocated as a result of their *relationship* to another *ancestor Concept*. However, the path from any *Concept* to the root *Concept* always converges at or before the top-level *Concept*. Therefore, multiple ranges coalesce when reaching more general common *supertype ancestors*.
- At run time, rather than needing to traverse many *subtype Relationships*, the branch number of each *Concept* is tested for inclusion in the branch number range of the putative *ancestor*.

This approach removes the need for exhaustive testing of *subtype Relationships*. The disadvantages are a relatively complex build process that must be repeated for each release or update and a requirement for the internal *Concept* representation to accommodate a variable length representation of branch number ranges.

7.7.5.1.5 Precomputed Transitive Closure table



The *transitive closure* table is a comprehensive view of all the supertypes of every *concept*. It can be derived from *current* release data by traversing all *| is a | relationships* recursively and adding each inferred supertype *relationship* to a table.

The advantage of this type of view is that a *candidate - concept* can be tested for subsumption by *predicate - concept* by a simple SQL query. In addition, the table can be updated to take account of changes without requiring a complete rebuild. The disadvantage is the storage capacity required.

 **Note:** The *transitive closure* table for the active content of the current version of the *International Release*, has about six million rows. The row count increases when *Extensions* are included. Typical database representations of the *transitive closure* table and associated indexes consume more than a Gigabyte of disk storage.

7.7.5.1.6 Recommendations



The *Transitive Closure* method is strongly recommended for use in any environment requiring high performance where disk capacity for storage and/or bandwidth for distribution are not a problem.

Where disk capacity and/or distribution bandwidth are limiting factors, Branch Numbering provides an efficient alternative approach.

7.7.5.2 Transitive closure implementation



Technology

The technology used to develop an *SNOMED CT enabled application* or used to query *SNOMED CT* data will affect the selection of the best implementation technique for the *transitive closure*.

If the *transitive closure* will be used to support SQL queries, a full *transitive closure* table needs to be created and stored *as a table in the relational database*.

In the cases where the transitive close will support actions in a software API, testing subsumption between in-memory objects, an *in-memory map* provides the best benefits.

7.7.5.2.1 Transitive closure distribution



It has been proposed that a *transitive closure* table should be released. This would support easier implementation and provide a reference against which to check alternative algorithms. The *transitive closure* table in a *full release* would contain a full history of the *transitive closure* since the first release of *SNOMED CT*. This would allow subsumption queries to be applied based on any release.

At present this table is not distributed and the format for such a distributed *transitive closure* table remains under discussion.

The following sub-sections provide basic advice on generating and using a simple and functional *transitive closure* table. Even if the *SNOMED CT International Edition transitive closure* is distributed, implementers may need to generate *transitive closures* including the content from one or more *Extensions*.

7.7.5.2.2 Transitive closure implementation in a relational database



7.7.5.2.2.1 Generating a transitive closure table



There are various ways in which a *transitive closure* table can be generated. The method illustrated here represents the smallest SQL query that might be used for this purpose. It may not be the most efficient query but on a typical Windows PC generates a snapshot *transitive closure* in about 5 minutes.

Table 225: MySQL script to Create a Snapshot Transitive Closure Table

```

-- SNOMED CT Transitive Closure for the Active Snapshot      --
-- Author: David Markwell 2010-2011                         --
-- Complete build of TC table for the most recent RFF2 version --
-- Takes 5 minutes to run on typical system                  --

-- ASSUMPTIONS
-- 1. Use of MySQL
-- 2. Database called `rf2` exists (or changed Initialize USE command)
-- 3. Database contains a table or view called `soa_relationship`
-- 4. The table `soa_relationship` contains an active snapshot (static or dynamic)
-- of the sct_relationship file(s) (including any extensions)
-- 5. The output table sct2_transitiveclosure contains the snapshot transitive
-- closure after completion.

-- Note: The soa_relationship view created by other sample scripts in this
-- document was used for testing this script.

-- Initialize database connection
USE rf2;

-- Set delimiter to allow procedure creation
DELIMITER $$

-- Create procedure to make the TransitiveClosure
DROP PROCEDURE IF EXISTS `sct2_make_tc`$$

CREATE PROCEDURE `sct2_make_tc`()
BEGIN
-- Initialise by removing existing tables
DROP TABLE IF EXISTS `sct2_transitiveclosure`;
DROP TABLE IF EXISTS `tmp_tc1`;
DROP TABLE IF EXISTS `batch_monitor`;

-- Create a table to allow batch process to be monitored (optional)
CREATE TABLE `batch_monitor` (
`step` int(11) NOT NULL,
`time` datetime DEFAULT NULL,
`recs` int(11) DEFAULT NULL,
`info` varchar(45) COLLATE latin1_general_cs DEFAULT NULL,
PRIMARY KEY (`step`)
);

-- Set the snapshot version time
SET @effectiveTime=configTime();

-- Initialize step counter
SET @step=0;

-- Record progress in batch_monitor table
INSERT INTO `batch_monitor`
(`step`, `time`, `recs`, `info`)
VALUES(@step,NOW(),0,'start');

-- Create empty sct_transitive closure table
CREATE TABLE `sct2_transitiveclosure` (
`subtypeld` BIGINT(20) NOT NULL ,
`supertypeld` BIGINT(20) NOT NULL ,
`effectiveTime` DATETIME,
`active` BOOLEAN,

```

```

PRIMARY KEY (`subtypeld`, `supertypeld`, `effectiveTime`),
KEY `ix_tc_main` (`subtypeld`, `supertypeld`),
KEY `ix_tc_inv` (`supertypeld`));

-- Create temporary first level transitive closure table
CREATE TEMPORARY TABLE `tmp_tc1` (
`subtypeld` BIGINT(20) NOT NULL ,
`supertypeld` BIGINT(20) NOT NULL ,
PRIMARY KEY (`subtypeld`, `supertypeld`),
KEY `ix_tc1` (`supertypeld`));

-- Insert Values into First Level TC
INSERT IGNORE INTO `tmp_tc1`(`supertypeld`, `subtypeld`)
SELECT `destinationId`, `sourceId` FROM `soa_relationship`
WHERE `active` =1 AND `typeId` = 116680003;

-- Create Level A temporary table for first iteration
DROP TABLE IF EXISTS `tmp_tcA`;

CREATE TEMPORARY TABLE `tmp_tcA` (
`subtypeld` BIGINT(20) NOT NULL ,
`supertypeld` BIGINT(20) NOT NULL ,
PRIMARY KEY (`subtypeld`, `supertypeld`),
KEY `ix_tc2` (`supertypeld`));

-- Copy Level 1 in to Level A for first iteration
INSERT IGNORE INTO `tmp_tcA`(`supertypeld`, `subtypeld`)
SELECT `supertypeld`, `subtypeld` FROM `tmp_tc1`;

-- Start the Loop each pass adds 2 steps to the semantic distance
TcLoop:LOOP
BEGIN
-- Increment the step count
SET @step=@step+1;

-- Count records in Level A
SET @rcount=(SELECT count(`supertypeld`) FROM `tmp_tcA`);

-- Batch monitor report (optional)
INSERT INTO `batch_monitor`
(`step`, `time`, `recs`, `info`)
VALUES(@step,NOW(),@rcount,'tcA');

-- If Level A empty then quit here
IF @rcount=0 THEN LEAVE TcLoop; END IF;

-- Append Level A records to final TC table
INSERT IGNORE INTO `sct2_transitiveclosure`(`supertypeld`, `subtypeld`, `effectiveTime`, `active`)
SELECT `supertypeld`, `subtypeld`, @effectiveTime, 1 FROM `tmp_tcA`;

-- Create Level B temporary table for this iteration (adds 1 to semantic distance)
DROP TABLE IF EXISTS `tmp_tcB`;

CREATE TEMPORARY TABLE `tmp_tcB` (
`subtypeld` BIGINT(20) NOT NULL ,
`supertypeld` BIGINT(20) NOT NULL ,
PRIMARY KEY (`subtypeld`, `supertypeld`),
KEY `ix_tc3` (`supertypeld`));

-- Insert A+1 into B
INSERT IGNORE INTO `tmp_tcB`(`supertypeld`, `subtypeld`)
SELECT `t`.`supertypeld`, `t1`.`subtypeld`
FROM `tmp_tcA` `t` INNER JOIN `tmp_tc1` as `t1`
```

```

ON `t`.`subtypeld`='t1`.`supertypeld'
LEFT OUTER JOIN `sct2_transitiveclosure` AS `tc`
ON `t`.`supertypeld`='tc`.`supertypeld' AND `t1`.`subtypeld`='tc`.`subtypeld'
WHERE `tc`.`subtypeld` is null;

-- Level B empty then quit here
SET @step=@step+1;

-- Level A empty then quit here
SET @rcount=(SELECT count(`supertypeld`) FROM `tmp_tcB`);
INSERT INTO `batch_monitor`
(`step`, `time`, `recs`, `info`)
VALUES(@step,NOW(),@rcount,'tcB');
IF @rcount=0 THEN LEAVE TcLoop; END IF;

-- Append Level B to final TC table
INSERT IGNORE INTO `sct2_transitiveclosure`(`supertypeld`, `subtypeld`, `effectiveTime`, `active`)
SELECT `supertypeld`, `subtypeld`, @effectiveTime, 1 FROM `tmp_tcB`;

-- Create Level A temporary table for next iteration
DROP TABLE IF EXISTS `tmp_tcA`;
CREATE TEMPORARY TABLE `tmp_tcA` (
`subtypeld` BIGINT(20) NOT NULL,
`supertypeld` BIGINT(20) NOT NULL,
PRIMARY KEY (`subtypeld`, `supertypeld`),
KEY `ix_tc3` (`supertypeld`)
);

-- Insert B+1 into A
INSERT IGNORE INTO `tmp_tcA`(`supertypeld`, `subtypeld`)
SELECT `t`.`supertypeld`, `t1`.`subtypeld`
FROM `tmp_tcB` `t` INNER JOIN `tmp_tc1` AS `t1`
ON `t`.`subtypeld`='t1`.`supertypeld'
LEFT OUTER JOIN `sct2_transitiveclosure` AS `tc`
ON `t`.`supertypeld`='tc`.`supertypeld' AND `t1`.`subtypeld`='tc`.`subtypeld'
WHERE `tc`.`subtypeld` is null;

END;
END LOOP;

END$$
-- END OF PROCEDURE CREATION

-- RUN THE CREATED PROCEDURE
CALL `sct2_make_tc`()$$

-- Delete the procedure
DROP PROCEDURE IF EXISTS `sct2_make_tc`$$

```

7.7.5.2.2 Transitive closure table structure



The simplest form for a *transitive closure* table has two columns labeled "Subtypeld" and "Supertypeld". Each of these columns has a datatype that supports the *SNOMED CT Identifier* and is populated by *concept identifiers*.

This simple table requires one unique index "Subtypeld+Supertypeld" and a secondary non-unique index by "Supertypeld" to allow efficient reversed lookup.

Additional columns may be included to optimize some extended functionality. For example:

- A flag to indicate rows that represent links between a *concept* and its proximal *primitive supertypes*.

- If *inactive concepts* are included in the table, a flag to indicate the nature of any Historical Association traversed.
- A semantic distance count indicating the number of direct | is a | relationship between the subtype and supertype. Although such a number has not absolute meaning it may be useful as a relative measure of proximity.
- An Identifier of the transitive closure row. This may be of value for maintaining history of changes to transitive closures between releases.

7.7.5.2.2.3 Using the transitive closure table to check subsumption



The following SQL queries illustrate ways to use a transitive closure table to test subsumption.

The queries here use a MySQL database with the snapshot transitive closure table built as using the script documented in [generating a transitive closure table](#). In practice , the SQL queries shown here will often be used as clauses in more complex queries allowing many candidates and predicates to be tested as a condition of retrieval in a single query.

```
SET @cptid=233604007;
```

```
SELECT `subtypeld`  
FROM `sct2_transitiveclosure`  
WHERE `supertypeld`=@cptid;
```

Figure 110: Return the Concept.id values of all subtype descendants of a specified concept

```
SET @cptid=233604007;
```

```
SELECT `supertypeld`  
FROM `sct2_transitiveclosure`  
WHERE `subtypeld`=@cptid;
```

Figure 111: Return the Concept.id values of all supertype ancestors of a specified concept

-- This illustration returns a text message indicating the semantic relationships between two concepts
-- Change the concept Id values here to test other concepts.

```
SET @cptidA=233604007;  
SET @cptidB=422588002;
```

```
(SELECT CONCAT(CONVERT(@cptidB, CHAR), IF(count(`subtypeld`)  
' is '' is NOT ','a subtype of ',CONVERT(@cptidA, CHAR))  
FROM `sct2_transitiveclosure`  
WHERE `supertypeld`=@cptidA and `subtypeld`=@cptidB)  
UNION  
(SELECT CONCAT(CONVERT(@cptidA, CHAR), IF(count(`subtypeld`)  
' is '' is NOT ','a subtype of ',CONVERT(@cptidB, CHAR))  
FROM `sct2_transitiveclosure`  
WHERE `supertypeld`=@cptidB and `subtypeld`=@cptidA)  
UNION  
(SELECT CONCAT(CONCAT(CONVERT(@cptidA, CHAR), IF(@cptidA=@cptidB,' is '' is NOT '  
,a the same as ',CONVERT(@cptidB, CHAR)))  
;
```

Figure 112: Test whether concept is a subtype of another candidate concept

-- This query looks for concepts that:
-- a) are subtypes of a specified concept; and
-- b) contain a term with a string matching a specified pattern.
-- Note: This query requires that the sct2_description table has a FULLTEXT index on `term`.
-- The soa_description view is derived from that table.

```
SET @cptid=71388002;  
SET @pattern='asthma';
```

```
SELECT `d`.`conceptId`, `d`.`term` FROM `soa_description` `d`
```

```
JOIN `sct2_transitiveclosure` `t` ON `d`.`conceptId` = `t`.`subtypedId`
WHERE MATCH (`d`.`term`) AGAINST (@pattern) AND `t`.`supertypeId` = @cptid;
```

Figure 113: Matching terms for subtypes of a specified concept

7.7.5.2.3 Transitive closure implementation in memory



For real-time *subsumption tests*, an in-memory map performs better than a lookup on a persisted table in a relational database.

Map Structure:

- Map key: *Subtype concept Concept Identifier*
- Map value: Collection of direct parents *Concept Identifiers*

The high speed provided by in-memory structures allow us to have a simpler *transitive closure* representation, including only the direct parents of the *concept* (not all the ancestors, like in the relational database approach), and with only one appearance of the *subtype concept* in the map.

A recursive algorithm will check subsumption for any pair of candidate ids, navigating the map, looking for the parents of the *subtype* candidate and iteratively for all the parents of the parents, until it reaches the *root concept* (*concept* with an empty parents collection on the map value). If the parent candidate is found in any of the parents collections during the recursive map navigation, then iteration stops and the *subsumption test* returns true.

This approach provides a very compact representation, a full *transitive closure* map occupies around 12 megabytes,

The map creation process is straightforward, a single iteration of all "*Is a*" *Relationships* would retrieve all the necessary information for the map. In an editing environment the update of the map is also very simple, having only the direct parents represented in the map, changes in one *concept* affect only one value of the map. If the implementation uses a DL Classifier, the whole map should be updated after a classification run.

7.8 Supporting Selective Data Retrieval



This section addresses the types of *terminology service* that are required to enable effective use of the *SNOMED CT* hierarchies and definitions when retrieving data.

The actual process of data retrieval is a *record service*, rather than a *terminology service* because it involves interactions with a database containing instance data (e.g. an *electronic health record* or data warehouse repository). However, queries may use predicates that specify *subtypes* of particular *concepts* or specify *concepts* that have particular defining *Relationships*. In order to resolve these queries, the application will need to provide or use *Terminology services*, adapted to the implementation model in use. The use of either a local extension with a "Managed content addition" strategy, or *postcoordinated expressions* strategy with an *expressions* reference table, has an effect on the required set of *Terminology services*. (see [How to choose a SNOMED CT extension strategy](#))

7.8.1 Creating queries



A *terminology server* should support the creation of queries that retrieve *SNOMED CT* encoded data by facilitating the generation of predicate statements.

For example, a *terminology server* may generate an SQL predicate list that includes the *Concept Identifiers* of all unique *subtype descendants* of a specified *Concept*. Some *constraints* on this functionality may be necessary as top-level or other general *Concepts* may generate extremely long lists of *descendant Concept Identifiers*.

7.8.2 Types of queries



There are different ways of representing a terminology query that can be sent to a *terminology server*:

- *Concept* and *Refset* references: lists of Ids of *concept* that can be retrieved from the server.
- Text based queries: text phrases that will be applied to *concept Descriptions* in order to retrieve results for the query.
- *Concept definition* queries: the query provides a *concept* definition, and the server returns all *concepts* that are subsumed by, or are equivalent to the definition.
- *Expression* retrieval: predicate terminology *expressions* that can be applied to all candidate *concepts* in the terminology to test for inclusion in the results, with filtering by hierarchy and attributes. These *expressions* are defined using a standard *expression* grammar that can be parsed and transformed in order to be evaluated against candidate *concepts*.
- Query languages: a query language combines any of the previous data retrieval techniques in the same syntax, including references to *concepts* by Ids, by text searches, by *refsets* or by hierarchy and attributes.

7.8.2.1 Concept and Refset references



The predicate for search for *Concepts* or *Refsets* includes references to one or more specific *concepts* by *Concept Identifier*, and additional instructions on how to retrieve related *concepts*.

Example use cases:

1. An application populates a combo box with a list of *concepts* pre-defined in a Simple Type *Refset*, identified by Id.
2. An HL7 message provides a reference to a *concept* by Id, the *preferred term* for the local implementation is retrieved based on the conceptId.
3. An application provides a tree view of the hierarchy, the *root concept* is referenced by Id and any level of children *concepts* are retrieved based on user selections.
4. A *concept* and all its *descendants* are retrieved in order to match with clinical records, where no *postcoordination* is used.

Example queries:

1. Retrieve the specified *Concept Identifier*
2. Retrieve direct *subtypes* (children) of the specified *Concept Identifier*
3. Retrieve *descendants* (children and their children recursively) of the specified *Concept Identifier*.
4. Retrieve members of the Simple Type *Refset* identified by the specified *Concept Identifier*

This kind of selective data retrieval can be easily implemented using SQL, *Concept Identifiers* are the primary key or foreign keys in all the necessary tables in the *SNOMED CT* model. Retrieval of *descendants* can be optimized as detailed in the "[Optimizing concept subsumption testing](#)" chapter.

Implementations based primarily on *precoordinated* content will be able to support most of their use cases with this kind of query. *SNOMED CT* content is distributed with a pre-computed inferred view, that can be trusted to retrieve all related *concepts* by references to a *concept* in the terminology.

Implementations that have created local extensions, will require the availability of a *Descriptions Logic* classifier in order to periodically compute a new inferred view that will discover new *relationships* between *concepts* in the terminology.

In use cases where clinical data is recorded as *postcoordinated expressions*, or where *postcoordinated expressions* are used as the query, the techniques described in the [Expression retrieval section](#) should be applied.

7.8.2.2 Text based queries



Support for Text based queries is a fundamental component of the *terminology server*,

 **Example use cases:**

1. A user enters a text string that is matched with *SNOMED CT* content in order to retrieve the most similar candidates
2. A mapping support tool finds closest matches in *SNOMED CT* for a local terminology or a classification, in order to provide lexical mapping suggestions to users.

This subject is discussed in the [User Interface Terminology Services section](#).



7.8.2.3 Concept definition queries

Concept definition queries allow the user to submit a *concept* definition to the *Terminology Server*, and the server returns all the *concepts* that are either equivalent or subsumed by the input *concept* definition.

 **Example use cases:** A Hospital gets an *concept* definition as part of a Decision Support Rule that is shared with another hospital. The hospital staff needs to find out what *concepts* in their terminology are equivalent or subsumed by that *concept* definition. They have implemented a Managed Content Additions (MCA) model, and their patient records include references to *precoordinated concepts* in their local terminology. They don't store *postcoordinated expressions* in the clinical record.

Possible representation formats are covered in the [Representational Forms](#) chapter, and they include *SNOMED CT Release Formats*, OWL, KRSS, *SNOMED CT postcoordinated expressions* and others.

A *Description logic classifier* is used to classify the *concept* definition with the candidate terminology, being the official distribution of *SNOMED CT* or an extension. The classifier will output the list of equivalent *concepts*, and will create an updated set of inferred *Relationships* that allow the detection of all *descendants concepts*.

However, if the server needs to match a repository of *postcoordinated expressions* the [Expression retrieval](#) technique becomes the required approach, transforming the *concept* definition into an *expression*.



7.8.2.4 Expression retrieval and normal forms

A *terminology server* should support selective retrieval by facilitating testing of *expressions* against *query predicates*.

Using *expressions* is the most common approach for supporting *postcoordination*.

 **Example use cases:**

1. Users define a *postcoordinated expression*, the server verifies if the *expression* is an exact match with an existing *precoordinated concept* or if it will be stored in a *postcoordinated expressions* repository as a new or existing *expression*.
2. For epidemiological purposes a predicate *expression* is created as a query. The predicate *expression* is matched against candidate *expressions* and *concept* references stored in the clinical record, to retrieve all content equivalent or subsumed by the predicate *expression*.

To facilitate complete and accurate retrieval of *precoordinated* and *postcoordinated expressions* from clinical records or other resources it is necessary to compare an *expression* in a record with a *query predicate*. This comparison needs to determine if the candidate *expression* is subsumed by the predicate.

The same meaning can be represented in different *postcoordinated expressions* and to facilitate comparison *expressions* with the same meaning can be converted to a common *normal form*. This section describes the process of normalization and the approach to testing for subsumption between the resulting *normal form expressions*.



7.8.2.4.1 Candidate and predicate expressions

In a *subsumption test* there are two *expressions*, one of which is being tested for subsumption by the other. To distinguish these *expressions* the following definitions are used:

Candidate expression - An *expression* that is being tested to see if it is subsumed by another *expression*.

Predicate expression - An expression that is being tested to see if it subsumes another expression.

Table 226: Example Predicate and Candidate Expressions

Predicate	Candidate	Test result
Fracture of femur	Fracture of neck of femur	True
	Fracture of bone	False
Fracture of bone	Fracture of femur	True
	Fracture of neck of femur	True
asthma (<i>in patient</i>)	FH: Asthma	False
	Severe asthma (<i>in patient</i>)	True
Family history of respiratory disease	FH: Asthma	True

7.8.2.4.2 Expression parts



The figures in this section illustrate some terms used to describe different parts of an expression in the discussion of [normal forms](#), the guidance on [transforming expressions to normal forms](#) and on [testing subsumption and equivalence between expressions](#).

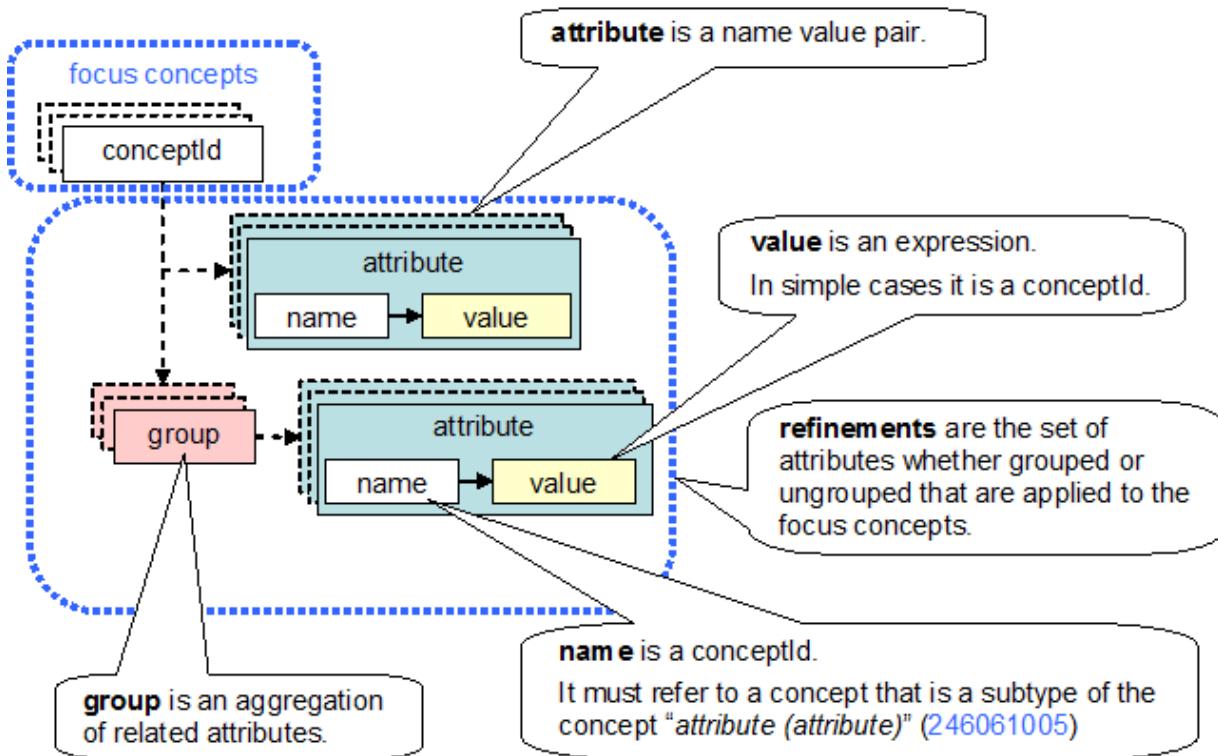


Figure 114: Focus concepts and refinements

As illustrated by [Figure 114](#), an *expression* consists of one or more conceptIds plus optional *refinements*. The *refinements* may include any number of attributes. *Attributes* are expressed as name-value pairs and may apply independently or as part of a group.

The name part of the *attribute name* -value pair is a conceptId that refers to a *concept* that names the characteristic that is refined by this attribute. The value part of the *attribute name* -value pair is an *expression*. In simple cases, this is simply a conceptId referring to a *concept* that represents the appropriate value for this attribute. However, it may also be a nested *expression* as shown in [Figure 115](#).

[Figure 115](#) illustrates the potential for nesting of *expressions* and the naming conventions applied in this guide to distinguish different parts of an *expression* at different levels. The top level of an *expression* is referred to as the "focus *expression*". It consists of a set of one or more "focus *concepts*" and a "focus *refinement*". The values of the attributes in the focus *refinement* are "nested *expressions*" that consist of one or more "value *concepts*" optionally refined by a "nested *refinement*".

Expressions may be nested recursively so there may be further levels of "nested *expressions*" with "nested *refinements*". If it is necessary to distinguish the level of nesting, the following naming convention is applied.

Table 227: Expression Nesting

Level number	Description
level 0 expression	Focus expression
level 1 expression	Nested expression
level N expression	An <i>expression</i> nested inside a level (N - 1) <i>expression</i>

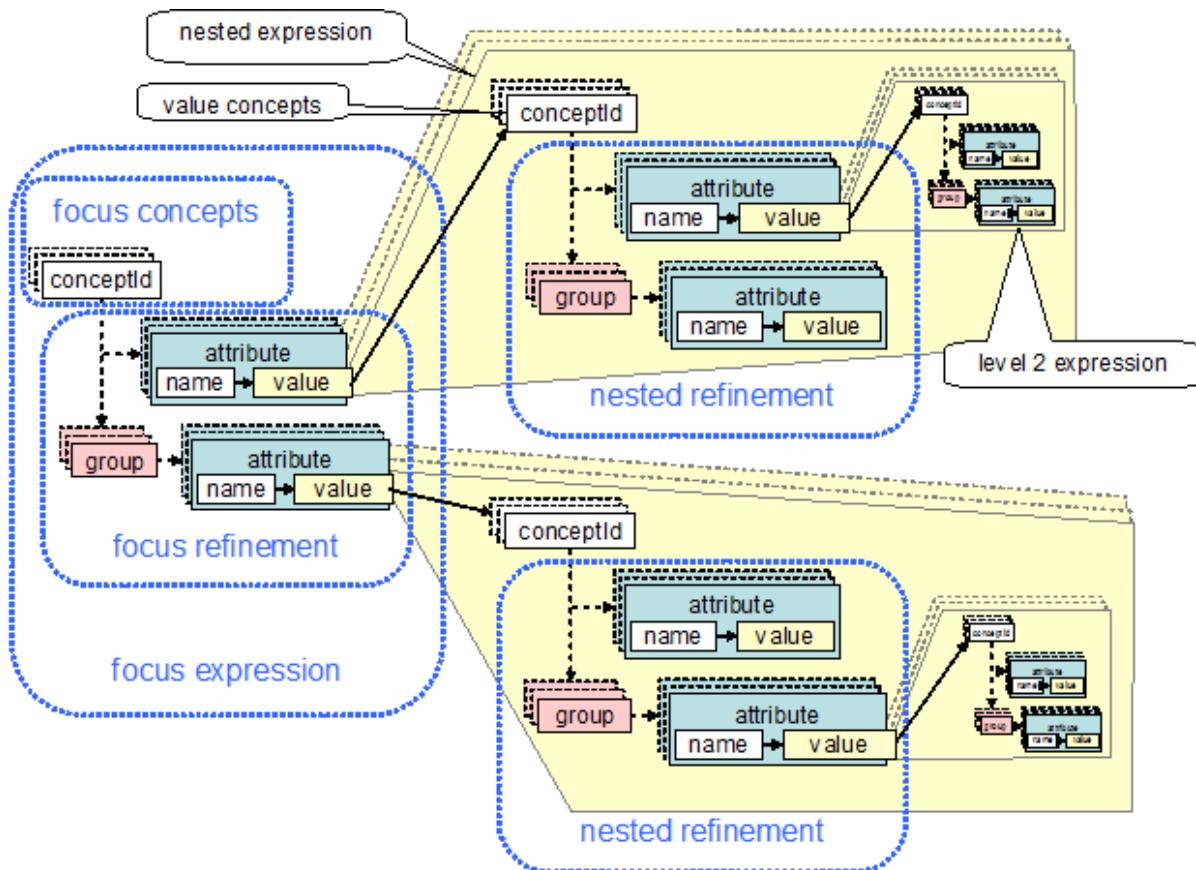


Figure 115: Illustration of the names used to refer to parts of a nested expression

The general pattern shown in [Figure 115](#) applies to all *expressions* whether or not they include *SNOMED CT* context information. [Figure 116](#) illustrates the specific features of an *expression* that includes a representation of *SNOMED CT* context.

The "focus expression" of a context containing expression is the "context wrapper" and may include a "context refinement" consisting of a set of context attributes:

- | associated finding | or | associated procedure | ;
- | finding context | or | procedure context | ;
- 408732007 | subject relationship context | ;
- | temporal context | .

In a normalized context expression, all context attributes are grouped. Each group in a normalized *context wrapper* contains a complete set of four context attributes ¹⁰.

The value of the | associated finding | or | associated procedure | is a "nested expression" which is referred to as the "clinical kernel".

During some stages of processing, the "clinical kernel" is separated from the "context wrapper". When separated from its context the "clinical kernel" is the "focus expression" of a context-free expression.

¹⁰ Usually a single group is present in a context expression. Theoretical cases exist for multiple groups where different contexts apply to different aspects of a concept but these cases are beyond the scope of the normalization rules in this guide.

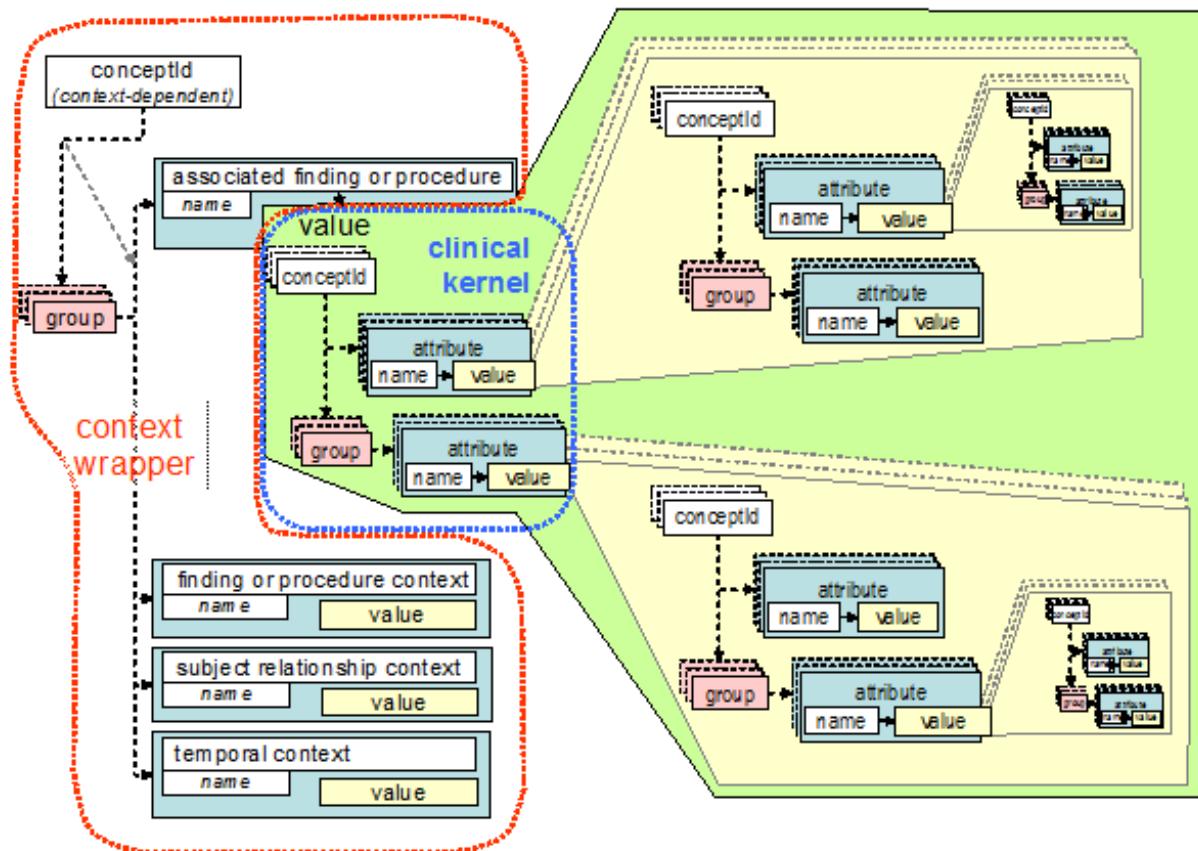


Figure 116: Illustration of the names used to refer to parts of an expression that represent context

7.8.2.4.3 Normal forms



A *normal form* is a view that can be generated for any valid *expression* by applying a set of logical *transformation rules*. Once converted to their *normal forms*, *expressions* can be more easily tested for subsumption by one another.

7.8.2.4.3.1 General characteristics of normal forms



All the conceptIds present in a *normal form expression* refer to *primitive concepts*. When normalizing an *expression*, every conceptId is replaced with the *normal form expression* that represents the definition of the referenced *concept*.

Normalization is recursive so that any element of a *concept* definition that refers to another *fully defined concept* is also replaced by the *normal form* of that *concept*.

One test of normalization is that applying the rules to an already normalized *expression* should return an identical *expression*.

7.8.2.4.3.2 Rationale for long and short normal forms



There are two distinct *normal forms* that are of value when computing subsumption.

The long *normal form* is appropriate for a candidate *expression* because it explicitly states all the attributes can be inferred from *concepts* referenced by the *expression*. This makes it easier to test whether the candidate fulfills a set of predicate conditions.

The short *normal form* is more appropriate for predicate *expressions*. It enables more efficient retrieval testing because there are fewer conditions to test. However, there is no loss of specificity because any candidate that fulfills the conditions of the short *normal form* inevitably fulfills the conditions of the long *normal form*.

7.8.2.4.3.3 Building long and short normal forms



The most effective approach to building either *normal form* is to start by generating the long *normal form*. If the short *normal form* is required this can then be derived by removing redundant defining *relationships*.

Generating a long form to derive short form may appear counterintuitive. However, there are three *reasons* why this approach is strongly recommended.

- The process of generating either *normal form* includes steps that test subsumption between different parts of an *expression*. The long *normal form* is required as the predicate for these tests.
- A single approach requires only one algorithm to be specified and implemented. This eases maintenance and reduces the risks of inconsistencies developing between the two *transforms*.
- The short form is needed less frequently than the long form because it is used in predicates (e.g. queries) rather than in candidate instances (e.g. *expression* in a record). An approach that optimizes long form generation is therefore advantageous.

The next section sets out the general approach to generation of the long and short *normal forms* for a *concept* definition. References to individual *concepts* in the source *expression* are replaced by *normal form concept* definitions when generating the *normal form* of an *expression*.

7.8.2.4.3.4 Concept definitions in normal forms



7.8.2.4.3.4.1 Long Normal Form



A form which when applied to a candidate *expression* allows effective computation of whether it is subsumed by a predicate *expression*.

Supertype view: **Proximal Primitive Supertypes**

- For *fully defined concepts* compute the proximal *primitives*
- For *primitive concepts* treat the *concept* itself as the proximal *primitive supertype*:
 - Rationale: This *primitive concept* must be present to enable the candidate *expression* to be subsumed by a predicate *expression* that includes this particular *primitive concept*.

Attribute view: **All Defining Relationships**

- For all *concepts* (whether *fully defined* or *primitive*) include all non - *subtype* defining *relationships*, irrespective of whether these are also present in the *union* of the definitions of the *primitive supertypes*:
 - Rationale: An *expression* may be subsumed by a *concept* that does not share all its proximal *primitive supertypes*. Some of the characteristics specified as part of other *primitives* in the candidate *expression* may also be present in the candidate *expression*.

7.8.2.4.3.4.2 Short Normal Form



A form which when applied to a predicate *expression* allows effective computation of whether a candidate *expression* is one of its *subtypes*.

Supertype view: **Proximal Primitive Supertypes**

- For *fully defined concepts* compute the proximal *primitives*
- For *primitive concepts* treat the *concept* itself as the proximal *primitive supertype*:
 - Rationale: As for long form see [Long Normal Form](#) on page 348.

Attribute view: **Differential Defining Relationships** (compared to supertype view)

- For *primitive concepts* there are no differential defining *relationships* because the *primitive concept* is its own proximal *primitive supertype*. Therefore in predicate *normal form* the attribute view is empty for *primitive concepts*.

- For *fully defined concepts* the differential form only includes defining *relationships*, and *relationship groups*, that are more specific than those present in the *union* of the definitions of the *primitive supertypes*:
 - Rationale: Each element in the predicate specifies an additional test to be applied to candidate *expressions*. However these additional tests are superfluous because:
 - The candidate *expression* cannot be subsumed by the predicate unless every candidate *primitive supertype* is subsumed by at least one predicate *primitive supertype*;
 - If this condition is met, then all defining *relationships* or *relationship groups* or the candidate *primitive supertypes* are inevitably also shared by the candidate *expression*.

7.8.2.4.3.4.3 Examples of normal form concept definitions

7.8.2.4.3.4.3.1 Normal form of a fully-defined concept with no intermediate primitives



The *concept* | fracture of femur | is *fully defined* and its proximal *primitive supertype* is a high-level *primitive*¹¹. This proximal *primitive* does not share any of the defining *relationships* of the *concept* itself. Therefore, the long and short *normal forms* of | fracture of femur | are identical because all its defining *relationships* differ from those of its *primitive supertype*.

Table 228: Normal form of a fully-defined concept with no intermediate primitives

Concept	71620000 fracture of femur
Definition (distributed)	116680003 is a = 64572001 disease {116676008 associated morphology = 72704001 fracture ,363698007 finding site = 71341001 bone structure of femur }
Long NF (candidate)	64572001 disease : {116676008 associated morphology = 72704001 fracture ,363698007 finding site = 71341001 bone structure of femur }
Short NF (predicate)	64572001 disease : {116676008 associated morphology = 72704001 fracture ,363698007 finding site = 71341001 bone structure of femur }

Using the long *normal form*

To test if | fracture of femur | is subsumed by another *expression* the long *normal form* is used as the candidate. If all the conditions of a predicate *expression* are satisfied by this candidate then | fracture of femur | is subsumed by this predicate.

- The *concept* | fracture of femur | is subsumed by any *normal form* predicate *expression* with a *focus concept* | disease | (or a supertype of "diseases" such as | clinical finding |) unless the predicate expression also has conditions that do not subsume | morphology | = | fracture | and | finding site | = | bone structure of femur |.

Using the short *normal form*

To test if | fracture of femur | subsumes another *expression* the short *normal form* is used as the predicate. Any candidate *expression* that satisfies all the conditions of this candidate is subsumed by | fracture of femur |

¹¹ A high-level primitive is a concept that is primitive and has no fully defined supertypes.

- The *concept* | fracture of femur | subsumes any *concept* that is a | disease | with a morphology subsumed by "fracture" and a | finding site | subsumed by | bone structure of femur |:
- The candidate *expression* | disease | with | morphology | = | fracture, open | and | finding site | = | structure of neck of femur | is thus subsumed.

7.8.2.4.3.4.3.2 Normal forms of a Primitive concept



The *concept* "asthma" is *primitive* so it is its own proximal *primitive supertype*. The long *normal form* therefore consists of the *concept* itself and all its defining *relationships*. The short *normal form* is simply the *concept* itself.

Table 229: Normal forms of a Primitive concept

Concept	195967001 asthma [Primitive]
Definition (distributed)	116680003 is a = 41427001 disorder of bronchus ,116680003 is a = 79688008 respiratory obstruction {116676008 associated morphology = 26036001 obstruction ,363698007 finding site = 955009 bronchial structure }
Long NF (candidate)	195967001 asthma : {116676008 associated morphology = 26036001 obstruction ,363698007 finding site = 955009 bronchial structure }
Short NF (predicate)	195967001 asthma

Using the long *normal form*

To test if "asthma" is subsumed by another *expression* the long *normal form* is used as the candidate. If all the conditions of a predicate *expression* are satisfied by this candidate then "asthma" is subsumed by this predicate.

- The *concept* "asthma" is subsumed by any *normal form* predicate *expression* with a *focus concept* that is "asthma" (or a supertype of "asthma" such as | disease | or | clinical finding |) unless the predicate *expression* also has conditions that do not subsume | morphology | = | obstruction | and | finding site | = | bronchial structure |.

Using the short *normal form*

To test if "asthma" subsumes another *expression* the short *normal form* is used as the predicate. Any candidate *expression* that satisfies all the conditions of this candidate is subsumed by | asthma |.

- The *concept* "asthma", only subsumes *expressions* that explicitly include a *focus concept* that is either "asthma" or a *subtype* of | asthma |:
- The candidate *expression* | disease | with | morphology | = | obstruction | and | finding site | = | bronchial obstruction | is not subsumed by | asthma |.

7.8.2.4.3.4.3.3 Normal form of a fully-defined concept with an intermediate primitive



The *concept* | allergic asthma | is *fully defined* but its proximal *primitive* supertype ("asthma") is an intermediate *primitive*¹². The long *normal form* consists of the proximal *primitive* supertype and all the defining *relationships* of | allergic asthma |. The short *normal form* is the same proximal *primitive* but the only *relationship* included is | due to | = | allergic reaction | as this is its only difference from the definition of the *primitive*.

Table 230: Normal form of a fully-defined concept with an intermediate primitive

Concept	389145006 allergic asthma
Definition (distributed)	116680003 is a = 195967001 asthma ,116680003 is a = 418168000 disorder due to allergic reaction ,42752001 due to = 419076005 allergic reaction {116676008 associated morphology = 26036001 obstruction ,363698007 finding site = 955009 bronchial structure }
Long NF (candidate)	195967001 asthma : 42752001 due to = 419076005 allergic reaction {116676008 associated morphology = 26036001 obstruction ,363698007 finding site = 955009 bronchial structure }
Short NF (predicate)	195967001 asthma : 42752001 due to = 419076005 allergic reaction

Using the long *normal form*

To test if | allergic asthma | is subsumed by another *expression* the long *normal form* is used as the candidate. If all the conditions of a predicate *expression* are satisfied by this candidate then | allergic asthma | is subsumed by this predicate.

- The *concept* | allergic asthma | is subsumed by any *normal form* predicate *expression* with a *focus concept* that is "asthma" (or a supertype of "asthma" such as | disease | or | clinical finding |) unless the predicate *expression* also has conditions that do not subsume | morphology | = | obstruction | and | finding site | = | bronchial structure | and | due to | = | allergic reaction |

Using the short *normal form*

To test if | allergic asthma | subsumes another *expression* the short *normal form* is used as the predicate. Any candidate *expression* that satisfies all the conditions of this candidate is subsumed by | allergic asthma |.

- The *concept* | allergic asthma |, only subsumes *expressions* that explicitly include a *focus concept* that is either "asthma" or a *subtype* of "asthma" and the attribute | due to | = | allergic reaction |:
 - The candidate *expression* | disease | with | morphology | = | obstruction | and | finding site | = | bronchial obstruction | and | due to | = | allergic reaction | is not subsumed by | allergic asthma |.

¹² An intermediate primitive is a concept that is primitive but which has fully-defined supertypes and subtypes.

7.8.2.4.3.4.3.4 Normal form of a fully-defined concept with fully-defined attribute values



The *concept* | neoplasm of right lower lobe of lung | is *fully defined* with a high-level proximal *primitive* (| disease |). The long *normal form* consists of the proximal *primitive* supertype and all the defining *relationships* of | neoplasm of right lower lobe of lung |. However, the value of the | finding site | attribute (| structure of right lower lobe of lung |) is itself *fully defined*. Therefore, this value is also transformed to *normal form* (| structure of lower lobe of lung | with | laterality | = | right |). The short *normal form* is the same because all of the defining *relationships* differ from those of the proximal *primitive* supertype.

The standard *SNOMED CT distribution format* does not support explicit nesting of definitions. The *normal forms* described in this require nesting and this is supported by the *SNOMED CT expression model*. As a result, the *normal forms* shown here differ from the distributed *relationships table* and from the *canonical table*.

Table 231: Normal form of a fully-defined concept with fully-defined attribute values

Concept	126716006 neoplasm of right lower lobe of lung
Definition (distributed)	116680003 is a = 126713003 neoplasm of lung {116676008 associated morphology = 108369006 neoplasm ,363698007 finding site = 266005 structure of right lower lobe of lung }
Long NF (candidate)	64572001 disease : {116676008 associated morphology = 108369006 neoplasm ,363698007 finding site = (90572001 structure of lower lobe of lung : 272741003 laterality = 24028007 right)}
Short NF (predicate)	Same as long form

Using the long *normal form*

To test if | neoplasm of right lower lobe of lung | is subsumed by another *expression* the long *normal form* is used as the candidate. If all the conditions of a *predicate expression* are satisfied by this candidate then | neoplasm of right lower lobe of lung | is subsumed by this predicate.

- The *concept* | neoplasm of right lower lobe of lung | is subsumed by any *normal form* *predicate expression* with a *focus concept* that is | disease | (or a supertype of | disease | such as | clinical finding |) unless the *predicate expression* also has conditions that do not subsume | morphology | = | neoplasm | and | finding site | = | structure of lower lobe of the lung | with | laterality | = | right |.

Using the short *normal form*

To test if | neoplasm of right lower lobe of lung | subsumes another *expression* the short *normal form* is used as the predicate. Any *candidate expression* that satisfies all the conditions of this candidate is subsumed by | neoplasm of right lower lobe of lung |.

- The *concept* | neoplasm of right lower lobe of lung |, only subsumes *expressions* that have a | finding site | that is the "right lower lobe of the lung". However, because this site is normalized an *expression* that postcoordinates the laterality and the site will also be subsumed.

7.8.2.4.3.5 Applying normal forms to expressions



The previous section described the manner in which *normal form expression transformations* are applied to *concept* definitions. In this section this approach is extended to cover *expressions* which may contain *refinements* or qualifications of the released *concepts*.

7.8.2.4.3.5.1 Normal form of a simple expression



The simplest *expression* consists of a reference to a single *concept* (i.e. a single conceptId with no *refinements*). The *normal forms* for this are the same as those for the *concept* definition. [Table 232](#) illustrates this using one of the examples used in the previous section.

Table 232: A simple expression with no refinements

Expression view	Expression
Close-to-user	71620000 fracture of femur
Normal-form (short or long)	64572001 disease : {116676008 associated morphology = 72704001 fracture ,363698007 finding site = 71341001 bone structure of femur }

7.8.2.4.3.5.2 Normal forms of expressions with refinements



If a *refinement* specifies a more specific (*subtype*) value for one of the defining *relationships* of the *focus concept*, the refined value simply replaces the value in the definition. The examples in [Table 233](#) and [Table 234](#) illustrate this for *refinements* to either site or the morphology.

[Table 235](#) extends this example to include *refinements* to both the morphology and site. In all these examples while the *refinements* were not grouped in the close-to-user form, the *transformation* groups the site and morphology. This occurs because the refined values are replacing the defining values that were grouped.

Table 233: An expression with a refinement to finding site

Expression view	Expression
Close-to-user	71620000 fracture of femur : 363698007 finding site 29627003 structure of neck of femur
Normal-form (short or long)	64572001 disease : {116676008 associated morphology = 72704001 fracture ,363698007 finding site = 29627003 structure of neck of femur }

Table 234: An expression with a refinement to the nature of the morphology

Expression view	Expression
Close-to-user	71620000 fracture of femur : 116676008 associated morphology = 134341006 displaced fracture

Expression view	Expression
Normal-form (short or long)	64572001 disease : {116676008 associated morphology =134341006 displaced fracture ,363698007 finding site = 71341001 bone structure of femur }

Table 235: An expression with a refinement to the morphology and site

Expression view	Expression
Close-to-user	71620000 fracture of femur : 116676008 associated morphology = 134341006 displaced fracture ,363698007 finding site = 71341001 bone structure of femur
Normal-form (short or long)	64572001 disease : {116676008 associated morphology =134341006 displaced fracture ,363698007 finding site = 71341001 bone structure of femur }

7.8.2.4.3.5.3 Normal forms of expressions with qualifiers



Table 236 shows the effect of applying a *qualifier Attribute* to a *concept*. As this is a *qualifier* it is not present in the definition and there is no indication whether this *qualifier* should be grouped with the site and morphology. Therefore, the *qualifier* remains ungrouped in the *normal form*.

Table 236: An expression with a qualifier applied to specify severity

Expression view	Expression
Close-to-user	71620000 fracture of femur : 246112005 severity = 24484000 severe
Normal-form (short or long)	64572001 disease : 246112005 severity = 24484000 severe {116676008 associated morphology = 72704001 fracture ,363698007 finding site = 71341001 bone structure of femur }

7.8.2.4.3.5.4 Normal forms of expressions with nested refinements



A *refinement* may be applied to a value in the *expression* rather than directly to the *focus concept*. Laterality is the most obvious example of a nested *refinement* and is used for all the illustrations in this section. However, nesting also occurs with other *expressions*, notably *expressions* that include explicit representations of context (see [Expressions that include context](#) and [Normal forms and the context model](#)).

Table 237 illustrates this by showing the effect of applying laterality to *refinement* to the finding site. The resulting *normal form* groups the finding site and its nested laterality *refinement*, with the morphology because this sub-expression is a valid *refinement* of the defined site.

Table 237: An expression with refinement of the laterality (nested with body structure)

Expression view	Expression
Nested refinement of laterality	71620000 fracture of femur : 363698007 finding site = (71341001 bone structure of femur : 272741003 laterality = 7771000 left)
Normal-form (short or long)	64572001 disease : {116676008 associated morphology = 72704001 fracture ,363698007 finding site = (71341001 bone structure of femur : 272741003 laterality = 7771000 left)}

7.8.2.4.3.5.5 Normal forms representations of laterality



Table 237 used laterality as an illustration of nested refinement. However, lateralized findings or procedures may be represented in several different ways.

As shown in *Table 238* applying laterality as a nested *refinement* to finding that has defined site requires restatement of the finding site (even though this value is unchanged). This redundancy is removed when the *expression* is transformed to its *normal form*.

Table 238: Nested laterality refinement

Expression view	Expression
Close-to-user	47933007 foot pain : 363698007 finding site = (56459004 foot structure : 272741003 laterality = 7771000 left)
Normal-form (short or long)	22253000 pain : 363698007 finding site = (56459004 foot structure : 272741003 laterality = 7771000 left)

Table 239 shows an alternative that is available for sites where there is a *concept* that is specific for lateralized body structure. In this example, the value | left foot | is a valid *refinement* of finding site because it is a *subtype* of | foot structure |.

The *concept* | left foot | is *fully defined* and includes the defining *relationship* | laterality | = | left |. Therefore, the *normal form* of this *expression* is identical to the nested laterality example.

Table 239: Alternative expression refinements representing lateralization

Expression view	Expression
Lateralized body structure value	47933007 foot pain : 363698007 finding site = 22335008 structure of left foot

Expression view	Expression
Normal-form (short or long)	22253000 pain : 363698007 finding site = (56459004 foot structure : 272741003 laterality = 7771000 left)

Table 240 illustrates an alternative that is available in a limited number of cases where a *concept* exists that precoordinates a finding with a lateralized finding site. The example shown here is artificial because the *concept* | pain in left foot | is not present in *SNOMED CT* and there are no plans to add such *concepts*. However, some *concepts* of this nature do exist and their definitions when transformed result in the same *normal form* as the *expression* shown in the earlier example.

Table 240: Laterality precoordinated in a finding

Expression view	Expression
precoordinated expression	<some-id> pain in left foot :
Normal-form (short or long)	22253000 pain : 363698007 finding site = (56459004 foot structure : 272741003 laterality = 7771000 left)

Table 241 shows a close-to-user form in which laterality has been applied directly to a finding. For the purposes of computing *equivalence* and subsumption the *concept model* always treats laterality as applying to body structures rather than directly to findings or procedures. However, a simple *transform* rule allows a close-to-user *expression* consisting of a finding with a direct laterality *refinement* to be normalized. This normalization rule specifies that the laterality *refinement* is applied to all lateralizable sites in the normalized *expression*. The end result of this *transform* is exactly the same *normal form* as results from other approaches. The same approach can be used for procedures.

Table 241: Laterality applied directly to a finding

Expression view	Expression
Close-to-user expression	47933007 foot pain : 272741003 laterality = 7771000 left
Normal-form (short or long)	22253000 pain : 363698007 finding site = (56459004 foot structure : 272741003 laterality = 7771000 left)

Conclusions on approaches to laterality

In principle, all four of the representations shown in *Table 242* are acceptable and all of them can be transformed to the same *normal form*. However, laterality is only *precoordinated* with a limited number of findings, procedures and body structures. Therefore, the only representations that provide comprehensive coverage are the direct form (3) and the nested *normal form* (4). Superficially the *normal form* seem most

appropriate but based on a more detailed the direct close-to-user form (3) is recommended for recording, storage and communication.

Table 242: Alternative representations of laterality

Expression view	Expression	Applicability
1. precoordinated	<some-id> pain in left foot :	Limited Only available if a <i>concept</i> exists to represent the finding or procedure at a specific lateralized site.
2. Lateralized body structure value	47933007 foot pain : 363698007 finding site = 22335008 structure of left foot	Limited Only if available if a <i>concept</i> exists to represent the lateralized body structure.
3. Laterality applied directly to finding	47933007 foot pain : 272741003 laterality = 7771000 left	Simple general purpose close-to-user form.
4. Normal-form (short or long)	22253000 pain : 363698007 finding site = (56459004 foot structure : 272741003 laterality = 7771000 left)	Common form derived by transformation.

The direct close-to-user form (3) has three significant advantages when compared to the *normal form*:

- Where multiple sites are involved (see [Table 246](#)) or where multiple separately grouped actions apply to the same site, this approach avoids the need to specify laterality separately for each site¹³. Routinely presenting users with a choice of which sites are to be lateralize is likely to hinder acceptance.
- The nested approach "locks-in" the site value(s) and groupings present in the definition at the time the *expression* is authored. If a future release enhances or corrects that definition, instances of the same *refinement* before and after a change will not compute as equivalent. However, if laterality is applied directly, the derived *normal forms* will be identical irrespective of when the *expression* was created.
- The resulting *expressions* are simpler and more compact and the *transform* rules mean no information is lost.

7.8.2.4.3.5.6 Normal forms of expressions including refinements of a Primitive concept



When the *focus concept* is *primitive* or has an intermediate *primitive* supertype the normal and close to user forms are less likely to be the same as one another.

[Table 243](#) illustrates the effects of a *refinement* applied to *primitive concepts*. The same general rules apply but after the *transformation* process the same *primitive focus concept* remains. The short *normal form expression* is identical to the close-to-user form because the *refinement* represents the only difference between the long *normal form* and the definition of the *focus concept*.

¹³ Only on very rare occasions will a single finding or procedure require separate lateralization of different sites in its definition. However, support for the direct approach does not preclude the nested approach if it is necessary to associate different laterality refinement with different structures.

Table 243: An expression that refines a Primitive concept

Expression view	Expression
Close-to-user	12529006 expiratory crackles : 363698007 finding site = 303549000 entire lower lobe of lung
Normal-form (long)	12529006 expiratory crackles : 363698007 finding site = 303549000 entire lower lobe of lung ,363714003 interprets = 78064003 respiratory function ,418775008 finding method = (315306007 examination by method : {260686004 method = 129436005 auscultation - action ,363704007 procedure site = 257728006 anatomical concepts })
Normal-form (short)	12529006 expiratory crackles : 363698007 finding site = 303549000 entire lower lobe of lung

Table 244 illustrates the effects of a *refinement* applied to a *concept* with an intermediate *primitive supertype*. The short *normal form* contains the "due to" *Attribute* because this differs between the *focus concept* (allergic asthma) and the *primitive supertype* (asthma) as when as the | causative agent | specified in the *refinement*.

Table 244: An expression that refines a concept with an intermediate primitive supertype

Expression view	Expression
Close-to-user	389145006 allergic asthma : 246075003 causative agent = 260147004 house dust mite
Normal-form (long)	195967001 asthma : 246075003 causative agent = 260147004 house dust mite ,42752001 due to = 419076005 allergic reaction {116676008 associated morphology = 26036001 obstruction ,363698007 finding site = 955009 bronchial structure }
Normal-form (short)	195967001 asthma : 246075003 causative agent = 260147004 house dust mite ,42752001 due to = 419076005 allergic reaction

7.8.2.4.3.5.7 Normal forms for refinements of concepts with more complex definitions



Some *concept* definitions include multiple instances of the same defining attribute. Usually these are grouped separately, for example to represent a procedure that examines one body structure and removes another. When *refinements* are applied to these *concepts* a question arises as to which value is to be refined. In most cases, the *transform* rules allow this to be determined without requiring the close-to-user expression

to explicitly state the instance that is being refined. The *transform* rule states that an ungrouped *refinement* applies to any instance of the appropriate attribute that subsumes it.

Table 245 shows the effect of this *transform* rule when the *refinement* of procedure site is a *subtype* of one of the defining site attributes but not of the other one. The appropriate value is refined and the other value is unchanged.

Table 245: An expression that refines one of two sites

Expression view	Expression
Close-to-user	116028008 salpingo-oophorectomy : 363704007 procedure site = 280107002 entire left fallopian tube
Normal-form (short or long)	71388002 procedure : {260686004 method = 129304002 excision - action ,363704007 procedure site = (181463001 entire fallopian tube : 272741003 laterality = 7771000 left)} {260686004 method = 129304002 excision - action ,363704007 procedure site = 15497006 ovarian structure }

Table 246 shows a case in which a *refinement* of laterality is applicable to both the sites in the definition of a procedure (i.e. both | ovarian structure | and | fallopian tube structure | are lateralizable). Therefore, the resulting *normal form* shows this lateralization applied to both structures.

Table 246: An expression that lateralizes multiple sites

Expression view	Expression
Close-to-user	116028008 salpingo-oophorectomy : 272741003 laterality = 7771000 left
Normal-form (short or long)	71388002 procedure : {260686004 method = 129304002 excision - action ,363704007 procedure site = (15497006 ovarian structure : 272741003 laterality = 7771000 left)} {260686004 method = 129304002 excision - action ,363704007 procedure site = (31435000 fallopian tube structure : 272741003 laterality = 7771000 left)}

In a few cases, a *refinement* that is valid for more than one attribute may need to be applied specifically to just one of those attributes. In these cases, the close to user form should include an *attribute group* with at

least one other attribute from the appropriate group in the *concept* definition. This allows the distinction to be made by the *transform* rules for *attribute group* merging.

Otherwise, the close-to-user form should not repeat groups or additional attributes that are unchanged from the definition. These attributes and groups are derivable by the *normal form transformation*. However, if they are included in the stored or communicated close-to-user form they are "locked-in", which may impair *equivalence* testing across releases.

7.8.2.4.3.5.8 Normal forms and the context model



The *SNOMED CT* context model is designed to allow *equivalence* and *subsumption* testing to take account of difference in the context in which a finding or procedure *concept* is used. The same general *transform* rules apply to *concepts* that include explicit statements of context. However, in addition to these rules the default context to a finding or procedure *expression* that has no explicitly stated context. This additional step allows the *equivalence* and *subsumption tests* to be applied in exactly the same way to *expressions* without stated context and to those with a stated context.

Table 247 shows an *expression* in which the *focus concept* | family history of disorder | has a definition that includes stated context. The disorder | allergy to nuts | is stated as the | associated finding |. Both these *concepts* are transformed to their respective *normal forms*

- The *normal form* of | family history of disorder | is a context wrapped in which "person in the family" is the value of "subject relationships context";
- The *normal form* of | allergy to nuts | (106190000 | allergy | with | causative agent | = | nut |) becomes the nested value of the | associated finding | *Attribute* of the *context wrapper*.

Table 247: An expression that includes specific context information

<i>Expression view</i>	<i>Expression</i>
Close-to-user	281666001 family history of disorder : 246090004 associated finding = 91934008 allergy to nuts
Normal-form (short or long)	243796009 context-dependent category : {246090004 associated finding = (106190000 allergy : 246075003 causative agent = 13577000 nut) ,408729009 finding context = 410515003 known present ,408731000 temporal context = 410512000 current or specified ,408732007 subject relationship context = 303071001 person in the family }

Table 248 shows an *expression* that does not state its context. Applying the *transform* rules the *normal form expression* is generated as in previous examples. When the additional step to apply the default context is carried out a default context-wrapper (| known present | in "the subject of the record" at "current or specified time |") is created and the clinical *expression* becomes the clinical-kernel within this wrapper.

In this case, the morphology and finding site *Attributes* are omitted as the values of these attributes are the same as those defined for the *Primitive concept* 195967001 | asthma |.

Table 248: Applying default context to an expression

Expression view	Expression
Close-to-user	195967001 asthma : 246112005 severity = 255604002 mild
Normal-form with context (long)	243796009 situation with explicit context : {246090004 associated finding = (195967001 asthma : 246112005 severity = 255604002 mild {116676008 associated morphology = 26036001 obstruction ,363698007 finding site = 955009 bronchial structure }) ,408729009 finding context = 410515003 known present ,408731000 temporal context = 410512000 current or specified ,408732007 subject relationship context = 410604004 subject of record }
Normal-form with context (short)	243796009 situation with explicit context : {246090004 associated finding = (195967001 asthma : 246112005 severity = 255604002 mild) ,408729009 finding context = 410515003 known present ,408731000 temporal context = 410512000 current or specified ,408732007 subject relationship context = 410604004 subject of record }

7.8.2.4.3.5.9 Normal forms that take account of the information model



When *expressions* are used in record systems or electronic communication there is often some surrounding contextual information that may affect the way in which the meaning of the *expression* should be interpreted. For example, a reference to a disease within a part of a record dedicated to "family history" should not be interpreted as a diagnosis of the patient. This is a complex area because many different information models and conventions may apply in different systems. However, some general rules have been identified and can be defined in relation to standard reference models (e.g. the *HL7 Version 3 Reference Information Model*).

The general rules are that contextual information apparent in the surrounding information model should be separated from the *SNOMED CT expression* before it is normalized . The *expression* is then transformed to a *normal form expression*. If the resulting *normal form* contains a *context wrapper*, this is separated from the clinical-kernel. A new *context wrapper* is derived by merging the information model context and any context stated in any original *context wrapper*. The *SNOMED CT* default context is only applied to fill in the gaps where neither the information model nor the original wrapper provides a definitive value for a context *Attribute*. The clinical-kernel is then nested in the new *context wrapper*.

The examples in this section cover two of the most common areas in which context from the information model affects the meaning of a contained *expression* in a predictable and processable way.

Table 249 illustrates the fact that when an *expression* representing a procedure exists in an information construct that represents a request, the default | procedure context | value | done | is overridden by the information model. Thus the resulting *normal form expressions* show the | procedure context | value "requested".

Table 249: Information model representation of context affecting default context

Expression view	Expression
Information model	<u>Request</u> <ul style="list-style-type: none"> Represented by the information model for example <i>HL7 Observation.moodCode="RQO"</i>. 113075003 creatinine measurement, serum
Close-to-user <i>with context</i>	400999005 procedure requested : 363589002 associated procedure =113075003 creatinine measurement, serum
Normal-form <i>with context</i> (long)	243796009 context-dependent category : 363589002 associated procedure = (252144003 252144003 : 116686009 has specimen = (123038009 specimen : 370133003 specimen substance = 67922002 serum) ,246093002 component = 15373003 creatinine) ,408730004 procedure context = 385644000 requested ,408731000 temporal context = 410512000 current or specified ,408732007 subject relationship context = 410604004 subject of record
Normal-form <i>with context</i> (short)	243796009 context-dependent category : 363589002 associated procedure =113075003 creatinine measurement, serum 408730004 procedure context = 385644000 requested ,408731000 temporal context = 410512000 current or specified ,408732007 subject relationship context = 410604004 subject of record

Table 250 illustrates that when the information model applies a value to a measurement procedure the resulting statement expresses a finding (i.e. the finding of a specific value as a result of that procedure).

If the requirement is to distinguish between requested and completed procedures, the default procedure context-wrapper could be applied. This would (correctly) assert that the procedure had been done (| procedure context | = | done |).

However, if the requirement is to distinguish between goals and actual measured values the default finding context-wrapper is more appropriate. This would indicate that this was a finding that was known to be present (| finding context | = | known present |).

Table 250: Measurement procedures with values assigned in the information model

Expression view	Expression
Information model	<p><u>Report</u></p> <ul style="list-style-type: none"> Represented by the information model for example <i>HL7 Observation.moodCode="EVN".</i> <p>113075003 creatinine measurement, serum </p> <p>Value = 16 g/L</p> <ul style="list-style-type: none"> Represented by the information model for example by <i>HL7 Observation.value.</i>
Normal-form <i>with context</i> (short)	<p>243796009 situation with explicit context :</p> <p>{246090004 associated finding = 113075003 creatinine measurement, serum ,408729009 finding context = 410515003 known present ,408731000 temporal context = 410512000 current or specified ,408732007 subject relationship context = 410604004 subject of record }</p> <p>Value = 16 g/L</p> <ul style="list-style-type: none"> Represented by the information model for example by <i>HL7 Observation.value.</i>

7.8.2.4.4 Transforming expressions to normal forms



The process of transforming an *expression* to a *normal form* is based on the *description logic* definitions of the *concepts* referenced by the *expression*. Using this approach, *expressions* that are authored, stored and/or communicated in a relatively informal close-to-user form are logically transformed into a common normalized form. In this normalized form it is possible to apply simple rules to test subsumption between *expressions*.

The simplest case of a valid close-to-user *expression* is a single *conceptId*, and the approach described can be applied to these simple *precoordinated expressions*, as well as to more complex *expressions* that include multiple *conceptIds* and *refinements* (*qualifiers*).

The approach to normalization may be applied to specific *expressions* but may also be extended to take account of contextual information derived from the information model in which the *expression* is situated. Therefore, the *normal form* may include *SNOMED CT* context information, even if this is not present in the initial *SNOMED CT expression*.

The algorithm extends earlier work on *canonical forms* as follow:

- Normalizes *fully defined* values within definitions or *expressions* producing nested *expressions* that are fully normalized .
- Merges *refinements* stated in an *expression* with definitional *relationships* present in the definitions of the *concepts* referenced by the *expression*:
 - The merge process takes account of *refinements* that may not be grouped or nested in a manner that precisely reflects the structure of a *current* (or future) *concept definition*;
 - This avoids the need to add, store and communicate potentially spurious detail from *current* definitions to the *expression* recorded by a user or software application.
- Takes account of context rules including default context and a preliminary approach to moodCode mapping and handling of procedures with values (present in algorithm but not yet easily visible in test environment).
- Supports *subsumption tests* that take account of finding specified with | known absent | finding context.

Figure 117 illustrates an overview of the process of normalization of an *expression*. Subsequent sections describe the processes shown in this diagram.

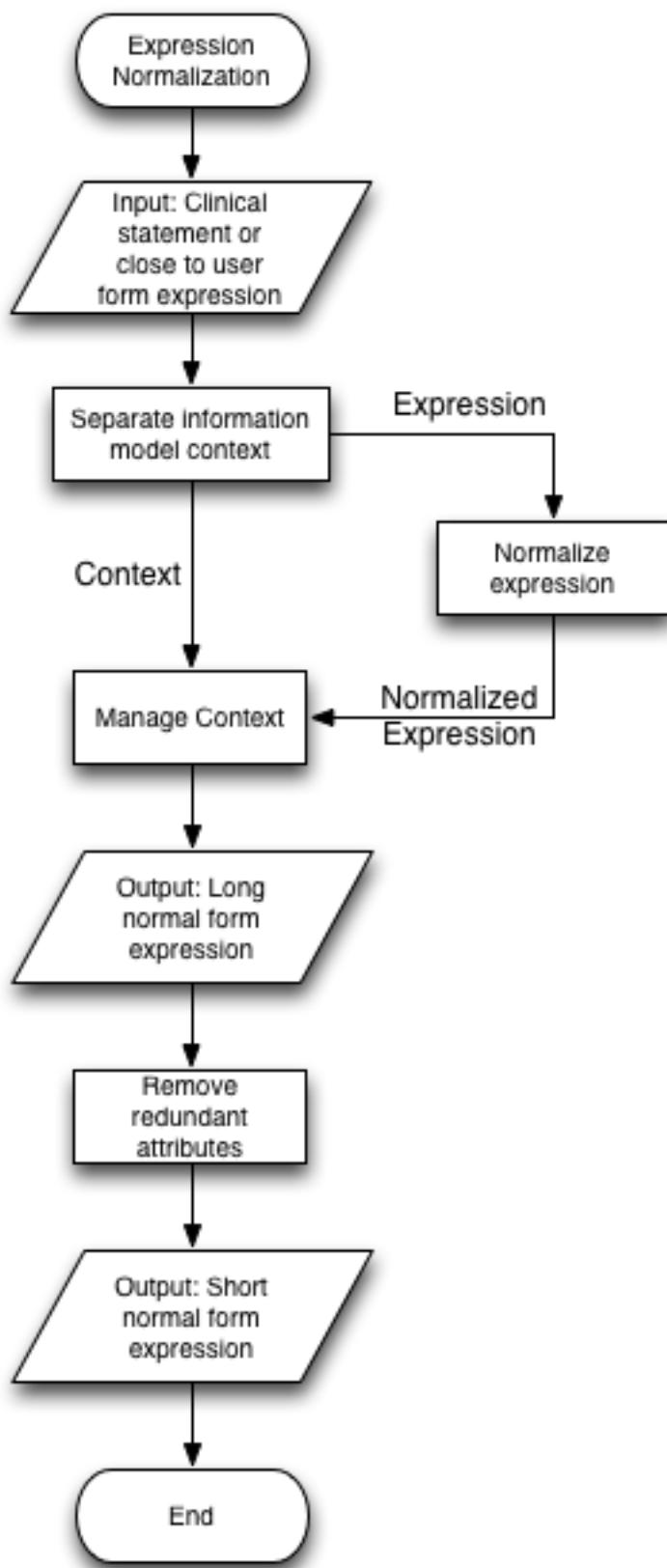


Figure 117: Overview of expression normalization process

7.8.2.4.4.1 Separate information model context



The objective of this process is to separate information associated with an *expression* from the *expression* itself.

Information that is not part of the *expression* itself, may influence its interpretation.

For example, a *expression* used in an *HL7* Observation in goal mood (*moodCode*="GOL") implies that the finding context | goal | applies to the *expression* rather than the default value | known present |.

If the input is a *expression* without any information about its use within a specific information model:

- The *expression* is passed unchanged to the "Normalize *expression*" process;
- No information model context is passed to the "Manage context" process.

If the input is an *HL7* clinical statement (or a similar structure that conveys additional contextual information):

- The *expression* is separated from the surrounding information model information and is passed to the "Normalize *expression*" process;
- Relevant surrounding information model information is passed to the "Manage context" process.

The items of surrounding information that are relevant vary according to the information model and the guidelines on its use. For example, if the *HL7* clinical statement model is used, any of the following attributes and related classes that are present are relevant to normalization of the *expression* in context:

- *Act.moodCode*;
- *Observation.value*;
- *Act.negationInd*;
- *Act.uncertaintyCode*;
- participation associations or an *Act* (especially "subject").

The way in which these attributes may affect *SNOMED CT* context is discussed in [Manage context](#).

Normalize expression



Figure 118 illustrates the detailed steps in the process of normalizing an *expression*. This process takes place after separating the *expression* from any surrounding information model context and before managing context representation in the *expression*.

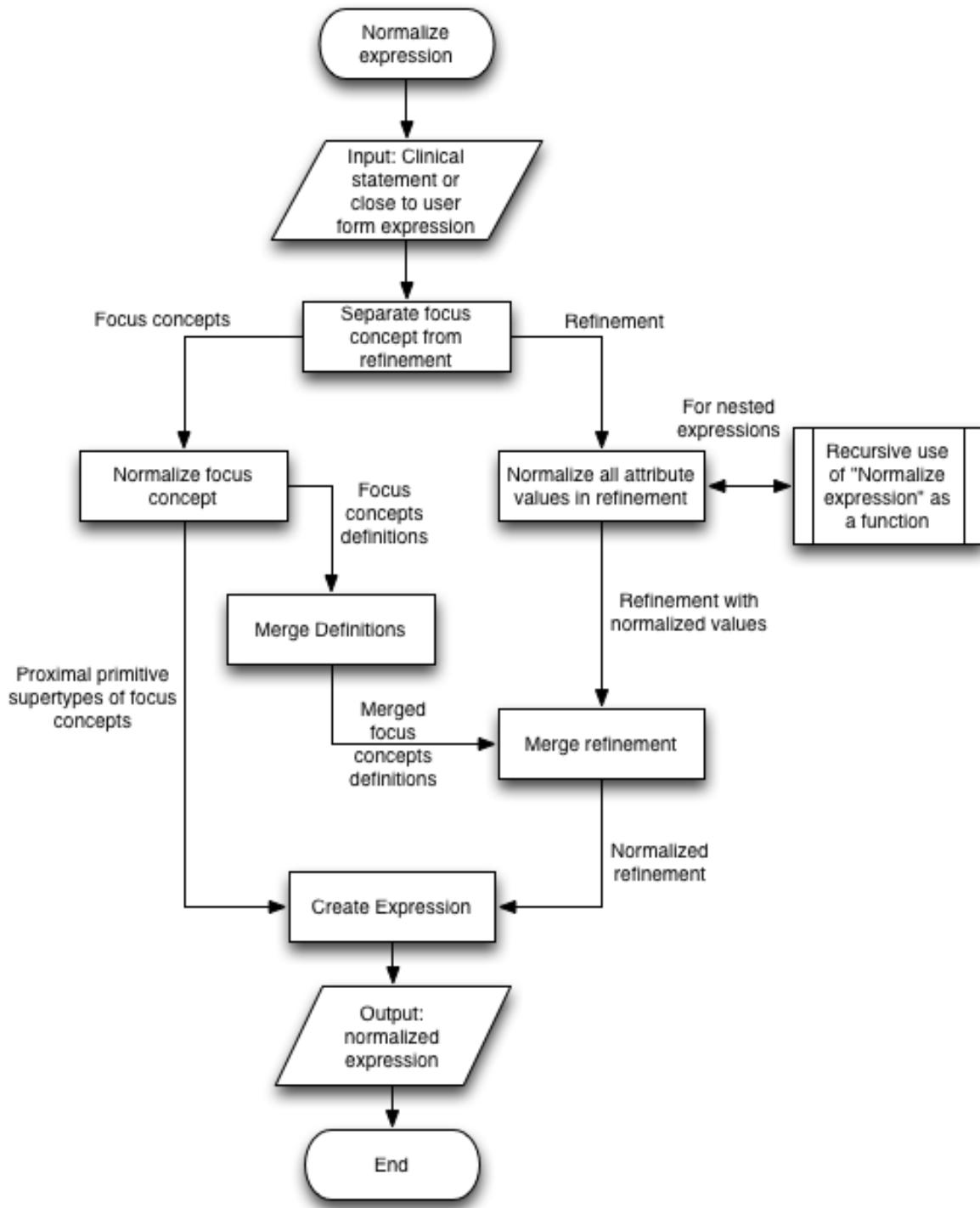


Figure 118: Expression normalization processes

7.8.2.4.4.2.1 Separate focus concepts from refinement



The set of *focus concepts* in the *expression* is passed to the [Normalize focus concepts](#) process.

If the *expression* contains a *refinement*, this is passed to the [Normalize attribute values in refinement](#) process.

Table 251: Separate focus concept from refinement

	Expression
Original expression	12676007 fracture of radius + 397181002 open fracture : 272741003 laterality = 7771000 left 42752001 due to = 297186008 motorcycle accident
Focus Concept	12676007 fracture of radius + 397181002 open fracture
Refinement	272741003 laterality = 7771000 left 42752001 due to = 297186008 motorcycle accident

Normalize attribute values in refinement



The value of every attribute specified in the *expression refinement* (including grouped and ungrouped attributes) is treated as an *expression* and normalized according to the full set of rules in [Normalize Normalise expression](#). To ensure depth-first processing, this recursive process is carried out before any other processing of the *expression refinement*.

Recursive normalization should be applied to all values even if they are represented by single conceptIds.

When all *attribute values* in the *expression refinement* have been processed, the *refinement* is passed to the [Merge refinement](#) process.

Normalize focus concepts



The set of *focus concepts* is normalized to generate two separate outputs described in the following sections.

7.8.2.4.4.2.3.1 The set of normalized definitions of each focus concept



The set of normalized definitions includes a separate normalized definition for each *focus concept*,

- The normalized definition includes:
 - All ungrouped *relationships*
 - All *relationship groups* complete with contained *relationships*
- All *relationship values* are normalized by recursively following the full set of rules described in [Concept definitions in normal forms](#).
 - 👉 **Note:** Storage of pre-computed normalized form of *concept definitions* simplifies this process as it removes the requirement for recursive processing of definitions at run time.

The set of normalized definitions is passed to the [Merge definitions](#) process.

Table 252: Normalize focus concepts definitions

	Expression
Original expression	12676007 fracture of radius + 397181002 open fracture : 272741003 laterality = 7771000 left 42752001 due to = 297186008 motorcycle accident

	Expression
<i>Focus Concepts</i>	12676007 fracture of radius + 397181002 open fracture
Set of normalized <i>focus concept</i> definitions	76069003 disorder of bone :{ 116676008 associated morphology = 72704001 fracture , 363698007 finding site = 23416004 bone structure of radius }+ 64572001 disease : 116676008 associated morphology = 52329006 open fracture

7.8.2.4.4.2.3.2 The non-redundant proximal primitive supertypes of the focus concepts



The non-redundant proximal *primitive* supertypes of the *focus concepts* is the set of all *primitive* supertypes of all the *focus concepts* with redundant *concepts* removed.

- A *concept* is redundant if it is:
 - A duplicate of another member of the set;
 - A supertype of another *concept* in the set.

The set of proximal *primitive* supertypes generated by this process is passed to the [Create expression](#) process as the *focus concepts* for the output expression.

Table 253: Normalize focus concepts definitions

	Expression
Original expression	12676007 fracture of radius + 397181002 open fracture : 272741003 laterality = 7771000 left 42752001 due to = 297186008 motorcycle accident
<i>Focus Concepts</i>	12676007 fracture of radius + 397181002 open fracture
Set of normalized <i>focus concept</i> definitions	76069003 disorder of bone :{ 116676008 associated morphology = 72704001 fracture , 363698007 finding site = 23416004 bone structure of radius }+ 64572001 disease : 116676008 associated morphology = 52329006 open fracture
List of all proximal <i>primitive</i> supertypes	76069003 disorder of bone 64572001 disease
List of non-redundant proximal <i>primitive</i> supertypes	64572001 disease

7.8.2.4.4.2.4 Merge definitions



The set of normalized definitions derived from the *Normalize focus concepts* process are merged with one another to remove redundancy. Then the normalized *refinement* is merged with the pre-merged definition to create a single *refinement* which expresses the full set of definitions and *refinements* without unnecessary redundancy.

The rules applied to the merger are described below for grouped and ungrouped attributes.

Group merging is completed before applying any ungrouped *relationships*. This ensures that, where appropriate, ungrouped attributes are applied to the correct groups in the output.

Redundant attributes are not removed until the merger process is complete. This ensures that the full set of attributes is available to allow matching throughout the process of merging.

7.8.2.4.4.2.4.1 Attribute names and attribute hierarchies



The following sections on merging groups and attributes refer to "name-matched" attributes.

Two or more attributes in a definition or expression are "name-matched" if they have the same *attribute name*¹⁴.

- For example, the attribute | procedure site | = | appendix structure | is name-matched by the attribute | procedure site | = | entire femur |.

However, consideration also needs to be given to hierarchical *relationships* between different "*attribute names*". For example, | procedure site - direct | and | procedure site - indirect | are *subtypes* of | procedure site |.

The simplest approach that can be consistently applied is to treat attributes that have subsumed names as name-matched for the purposes of group and value merging. The more specific *attribute name* is then applied to the merged attribute in the target definition. This means that the same rules apply for merging the values of | procedure site | and | procedure site - direct | as apply to mergers of attributes with identical names and that the name | procedure site - direct | would then be applied to any values that were merged in this way.

Progress note

Review of a number of practical examples suggests that there may be some unexpected consequences of this approach. For this *reason*, while the issues that arise are studied further, implementers are recommended only to merge literal name-matched attributes.

Some potential issues are noted here

As definitions are refined over time there will be more use of the specific | procedure site - indirect | and | procedure site - direct |. Should pre-existing *refinements* to the more general | procedure site | be assigned to whichever of the more specific attributes has a value that subsumes the refined value?

If this rule is applied to some combined procedures then the merger collapses some existing definitions that contain both a | procedure site | and a | procedure site - direct | so that only one of these attributes remains. This will become less of an issue as | procedure site - indirect | is applied more widely.

7.8.2.4.4.2.4.2 Merging groups



- If a group in one definition meets the following criteria in relation to a group in the other definition then the groups are merged:
 - At least one *attribute* in one of the groups is name-matched by an *attribute* in the other group.

and

¹⁴ The words "attribute" and "attribute name" are used here as documented in the SNOMED CT guide to the "Abstract Logical Models and Representation Forms". In SNOMED CT distribution files a "defining relationship" is equivalent to this use of the word "attribute" and a "relationship type" represents an "attribute name".

- For each name-matched pair of *attributes*, the value of that *attribute* in one group either subsumes or is identical to the value of the name-matched *attribute* in the other group;
- Groups that meet the criteria for merging are merged by adding all *attributes* present in both source groups to the same group in the merged target definition;
- Groups that cannot be merged are created as separate groups in the target definition.

Note that these conditions allow additional *attributes* that are not name-matched to be present in either of the candidate groups. They also allow values of name-matched *attributes* to be subsumed in different directions between the two groups (i.e. do not require the entire of one group to be subsumed the other group).

Table 254: Merging groups examples

Groups to merge	Result
<p><i>Group 1</i> 363704007 procedure site = 421235005 structure of femur , 363700003 direct morphology = 72704001 fracture </p> <p><i>Group 2</i> 260686004 method = 129371009 fixation - action , 424226004 using device = 31031000 Orthopedic internal fixation system, device </p>	<p>No match, no merge</p> <p><i>Group 1</i> 363704007 procedure site = 421235005 structure of femur , 363700003 direct morphology = 72704001 fracture </p> <p><i>Group 2</i> 260686004 method = 129371009 fixation - action , 424226004 using device = 31031000 Orthopedic internal fixation system, device </p>
<p><i>Group 1</i> 363698007 finding site = 62413002 radius , 116676008 associated morphology = 72704001 fracture </p> <p><i>Group 2</i> 363698007 finding site = 87342007 fibula , 116676008 associated morphology = 72704001 fracture </p>	<p>Attribute name match, but values don't match and are not subsumed in any direction ('radius' and 'fibula'), no merge</p> <p><i>Group 1</i> 363698007 finding site = 62413002 radius , 116676008 associated morphology = 72704001 fracture </p> <p><i>Group 2</i> 363698007 finding site = 87342007 fibula , 116676008 associated morphology = 72704001 fracture </p>

Groups to merge	Result
<p><i>Group 1</i> 363698007 finding site = 62413002 radius , 116676008 associated morphology = 72704001 fracture </p> <p><i>Group 2</i> 363698007 finding site = 87342007 distal radius , 116676008 associated morphology = 72704001 fracture </p>	<p>Attribute name match, 'distal radius' is subsumed by 'radius', merged groups.</p> <p><i>Group 1</i> 363698007 finding site = 62413002 radius , 116676008 associated morphology = 72704001 fracture </p> <p>363698007 finding site = 87342007 distal radius , 116676008 associated morphology = 72704001 fracture </p>

* Redundant elements will be removed later in the process

7.8.2.4.4.2.4.3 Merging ungrouped attributes



- If an ungrouped attribute in one definition is name-matched by a grouped attribute in the other definition, this attribute is merged according to the following rules:
 - If the value of the ungrouped attribute subsumes the value of the name-matched grouped attribute:
 - omit the ungrouped attribute from the target definition.
 - If the value of the grouped attribute subsumes the value of the name-matched grouped attribute:
 - add the ungrouped attribute to the group containing the matching grouped attribute in the target definition.
 - if this condition is met by multiple groups:
 - add the ungrouped attribute to all groups that meet this condition.
 - If the value of the name-matched grouped and ungrouped attributes are disjoint:
 - add the ungrouped attribute as an ungrouped attribute in the target expression.
- If an ungrouped attribute is name-matched with an ungrouped attribute in the other definition this attribute is merged according to the following rules:
 - If the value of one of the name-matched attributes subsumes the other value:
 - include the attribute with the most specific value (not grouped);
 - omit the attributed with the less specific value.
 - If the value of the name-matched attributes are identical:
 - Include one and omit the other.
 - If neither of the two preceding conditions apply:
 - include both attributes (not grouped).
- If an attribute is ungrouped in one *expression* and there is no name-matched attribute in the other definition:
 - include the attribute (not grouped).

Table 255: Merging ungrouped attributes

	Result
<p><i>Group 1</i> 363698007 finding site = 62413002 radius , 116676008 associated morphology = 72704001 fracture </p> <p><i>Group 2</i> 363698007 finding site = 87342007 distal radius , 116676008 associated morphology = 72704001 fracture </p> <p><i>Ungrouped</i> 42752001 due to = 297186008 motorcycle accident </p>	<p>Attribute name match, 'distal radius' is subsumed by 'radius', merged groups. Ungrouped attribute does not match. Not merged.</p> <p><i>Group 1</i> 363698007 finding site = 62413002 radius , 116676008 associated morphology = 72704001 fracture </p> <p>363698007 finding site = 87342007 distal radius , 116676008 associated morphology = 72704001 fracture </p> <p><i>Ungrouped</i> 42752001 due to = 297186008 motorcycle accident </p>
<p><i>Group 1</i> 363698007 finding site = 62413002 radius , 116676008 associated morphology = 72704001 fracture </p> <p><i>Group 2</i> 363698007 finding site = 87342007 distal radius , 116676008 associated morphology = 72704001 fracture </p> <p><i>Ungrouped</i> 116676008 associated morphology = 72704001 fracture </p>	<p>Attribute name match, 'distal radius' is subsumed by 'radius', merged groups. Ungrouped attribute matches name and value. Merged.</p> <p><i>Group 1</i> 363698007 finding site = 62413002 radius , 116676008 associated morphology = 72704001 fracture </p> <p>363698007 finding site = 87342007 distal radius , 116676008 associated morphology = 72704001 fracture </p> <p>116676008 associated morphology = 72704001 fracture </p>

* Redundant elements will be removed in later in the process

7.8.2.4.4.2.4.4 Remove redundant elements from the merged definition



Check each group in the target definition and, within that group, compare the values of any name-matched attributes.

- If an attribute in the group has a value that subsumes the value of another name-matched attribute in the same group, remove that attribute from this group in the target definition.

Check the ungrouped set of attributes.

- If any ungrouped attribute has a value that subsumes the value of a name-matched attribute, remove this ungrouped attribute from the target definition.

Note

The removal of redundancies described only applies to name-matched pairs of attributes. It does *not* affect attributes that are redundant *only* because they are present in the definitions of the *primitive focus concepts*. Supertype (| is a |) relationships are ignored during this stage of processing.

Table 256: Removing redundant elements

Merged definitions with redundancy	Redundancy removed
<p>Group 1</p> <p>363698007 finding site = 62413002 radius ,</p> <p>116676008 associated morphology = 72704001 fracture </p> <p>363698007 finding site = 87342007 distal radius ,</p> <p>116676008 associated morphology = 72704001 fracture </p> <p>116676008 associated morphology = 72704001 fracture </p>	<p>Group 1</p> <p>363698007 finding site = 87342007 distal radius ,</p> <p>116676008 associated morphology = 72704001 fracture </p>

7.8.2.4.4.2.4.5 Completion of the definition merging



Once the *focus concept* definitions have been merged, the target definition is passed to the *Merge refinement* process.

7.8.2.4.4.2.5 Merge refinement



The normalized *expression refinement* from the *Normalize attribute values in refinement* process is merged with the combined definition from the *Merge definitions* process. The rules for this process are the same as those for merging definitions.

Normalization of laterality



If an attribute representing a value for 272741003 | laterality | is present in the *refinement* and is applied to a *focus concept* that is not subsumed by 123037004 | body structure |, the laterality attribute should be applied to any and every lateralizable | body structure | specified in the resulting *refinement*.

Table 257: Normalization of laterality

	<i>Expression</i>
Original expression	12676007 fracture of radius : 272741003 laterality = 7771000 left
Normalized expression before laterality normalization	76069003 disorder of bone :{ 116676008 associated morphology = 72704001 fracture , 363698007 finding site = 23416004 bone structure of radius }, 272741003 laterality = 7771000 left

	Expression
Normalized expression after laterality normalization	76069003 disorder of bone :{ 116676008 associated morphology = 72704001 fracture , 363698007 finding site =(23416004 bone structure of radius , 272741003 laterality = 7771000 left)}{

Normalization of non-context attributes applied in a context wrapper



If the *focus concept* is subsumed by 243796009 | situation with explicit context | and any attributes other than valid context attributes¹⁵ are present in the *refinement*, these attributes are applied as additional *refinement* of the value of the 246090004 | associated finding | or 363589002 | associated procedure | attribute.

7.8.2.4.4.2.5.3 Completion of the definition merging



Once the *refinement* has been merged the resulting final *refinement* is passed to the *Create expression* process.

7.8.2.4.4.2.6 Create expression



The *create expression* process combines the proximal *primitive supertypes* from the *Normalize focus concepts* process (as the new *focus concepts*) - with the *refinement* derived from the *Merge refinement* process.

The resulting *expression* is now fully normalized but context information may need to be adjusted or applied by the *Manage context* process.

Table 258: Normalize focus concepts definitions

	Expression
List of non-redundant proximal <i>primitive supertypes</i>	64572001 disease
Normalized refinement without redundancy	<i>Group 1</i> 363698007 finding site = 87342007 distal radius , 116676008 associated morphology = 72704001 fracture
Resulting normalized expression	64572001 disease :{ 363698007 finding site = 87342007 distal radius , 116676008 associated morphology = 72704001 fracture }{

7.8.2.4.4.3 Manage context



Figure 119 illustrates the steps involved in managing context information extracted from the input statement or *expression*.

¹⁵ The only valid context attributes are: 246090004 | associated finding |, 363589002 | associated procedure |, 2470590016 | finding context |, 2470591017 | procedure context |, 2470592012 | temporal context | and 2470593019 | subject relationship context |.

The input to this process consists of the information model context, derived from the [Separate information model context](#) process, and the normalized expression, generated by the [Create Expression](#) step at the end of the [Normalize Normalise expression](#) process.

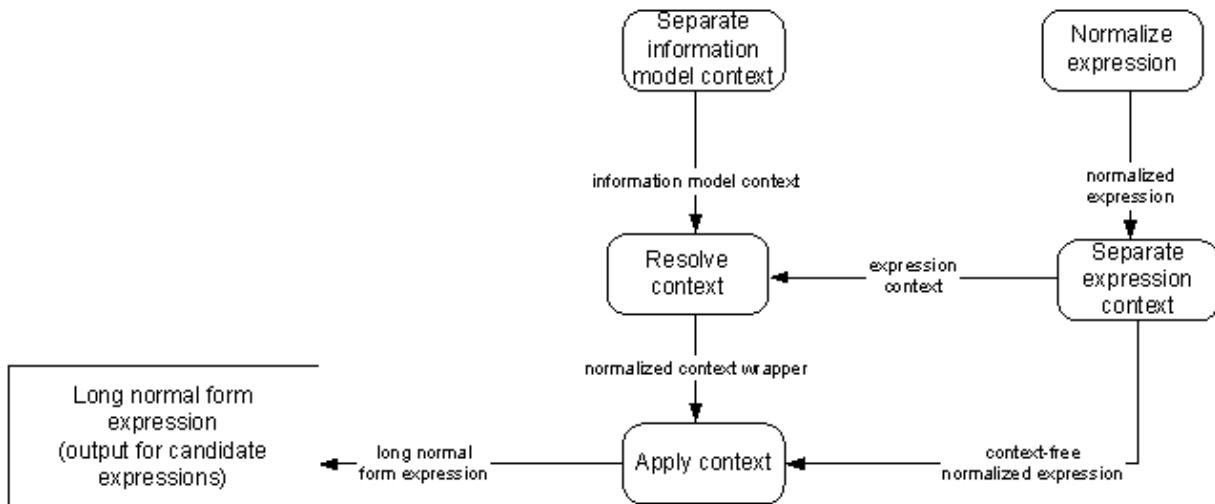


Figure 119: Managing context in normalized expressions

7.8.2.4.4.3.1 Separate expression context



The normalized expression (generated by the [Normalise expression](#) process) may or may not contain any context information. If it does, this context is separated from the expression so that it can be validated and reconciled with any information model context.

If the *focus concept* is subtype of 243796009 | situation with explicit context |

- The expression that represents the value of the | associated finding | or | associated procedure | attribute is passed as the context-free expression to the [Apply context](#) process;
- The focus expression without the | associated finding | or | associated procedure | attribute is passed to the [Resolve context](#).

If the *focus concept* is not a subtype of 243796009 | situation with explicit context | but its refinement contains values for one or more of the following context attributes: 2470590016 | finding context |, 2470591017 | procedure context |, 2470592012 | temporal context | or 2470593019 | subject relationship context |.

- These attributes are passed to the [Resolve context](#).
 - If the attributes present do not include a | finding context | or | procedure context | value, then an indication of the top level supertype of the *focus concept* is also passed with these context attributes.
- The focus expression, with the context attributes removed, is passed as the context-free expression to the [Apply context](#) process.

If neither of the above conditions apply then

- An indication of the top level supertype of the *focus concept* is passed to the [Resolve context](#) process;
- The entire expression is passed as the context-free expression to the [Apply context](#) process.

7.8.2.4.4.3.2 Resolve context



The resolve context process takes the information model context derived from the [Separate information model context](#) process and the expression context derived from [Separate expression context](#) process and attempts to resolve them to generate a single consistent context.

The context information in the expression or information model may unequivocally indicate that:

- Finding context applies:

- Subtypes of "finding" or "linkage concept";
- Subtypes of | procedure | or "observable" with an associated "value" in the information model;
- Finding context *attribute value* present in the *expression context* information.
- Procedure context applies:
 - Subtypes of | procedure | or "observable" without an associated "value" in the information model;
 - Procedure context *attribute value* present in the *expression context* information.

The appropriate default context applies unless modified

- By specific context attributes in the *expression context* information.
- By rules associated with particular information model context information:
 - For example, rules that in a reference file such as the MoodMap.xml (see [Figure 120](#)).

The output is one of the following

- A single *context wrapper* that is passed to the [Apply context](#) process.
- An indication that context is not relevant to the *expression* and should not be applied. This is also passed to the [Apply context](#) process allowing it to return a context-free *expression*.
- A report of errors arising from incompatibilities in the context information from the two sources.

```
<?xml version="1.0" encoding="UTF-8"?>
<!-- MoodMap.xml by David Markwell - Version 2005-03-26 -->
<!-- Copyright 2005 The Clinical Information Consultancy Ltd (www.clininfo.co.uk)
Licensed under the Apache License, Version 2.0 (the "License");
you may not use this file except in compliance with the License.
You may obtain a copy of the License at
```

<http://www.apache.org/licenses/LICENSE-2.0>

Unless required by applicable law or agreed to in writing, software
 distributed under the License is distributed on an "AS IS" BASIS,
 WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.
 See the License for the specific language governing permissions and
 limitations under the License.

-->

```
<!-- Suggested maps from HL7 Version 3 Mood Codes to default values for SNOMED Context attribute -->
<moodMap>
<moodCode code="-" term="Any">
<context id="408729009" term="finding context" defaultId="410515003" defaultTerm="known present"
permitted="410514004" permittedTerm="finding context value"/>
<context id="408730004" term="procedure context" defaultId="385658003" defaultTerm="done"
permitted="288532009" permittedTerm="context values for actions"/>
<context id="408731000" term="temporal context" defaultId="410512000" defaultTerm="current or specified"
permitted="410510008" permittedTerm="temporal context value"/>
<context id="408732007" term="subject relationship context" defaultId="410604004" defaultTerm="subject
of record" permitted="125676002" permittedTerm="person - add other permitted values in future"/>
</moodCode>
<moodCode code="EVN" term="Event">
<context id="408729009" term="finding context" defaultId="410515003" defaultTerm="known present"
permitted="410514004" permittedTerm="finding context value"/>
<context id="408730004" term="procedure context" defaultId="385658003" defaultTerm="done"
permitted="410523001" permittedTerm="post-starting action status"/>
<context id="408731000" term="temporal context" defaultId="410512000" defaultTerm="current or specified"
permitted="410510008" permittedTerm="temporal context value"/>
<context id="408732007" term="subject relationship context" defaultId="410604004" defaultTerm="subject
of record" permitted="125676002" permittedTerm="person - add other permitted values in future"/>
</moodCode>
<moodCode code="GOL" term="Goal">
```

```

<context id="408729009" term="finding context" defaultId="410518001" defaultTerm="goal"
permitted="410518001" permittedTerm="goal"/>
<context id="408731000" term="temporal context" defaultId="410512000" defaultTerm="current or specified"
permitted="410510008" permittedTerm="temporal context value"/>
<context id="408732007" term="subject relationship context" defaultId="410604004" defaultTerm="subject
of record" permitted="125676002" permittedTerm="person - add other permitted values in future"/>
</moodCode>
<moodCode code="INT" term="Intent">
<context id="408730004" term="procedure context" defaultId="410522006" defaultTerm="pre-starting action
status" permitted="410522006" permittedTerm="pre-starting action status"/>
<context id="408731000" term="temporal context" defaultId="410512000" defaultTerm="current or specified"
permitted="410510008" permittedTerm="temporal context value"/>
<context id="408732007" term="subject relationship context" defaultId="410604004" defaultTerm="subject
of record" permitted="125676002" permittedTerm="person - add other permitted values in future"/>
</moodCode>
<moodCode code="RQO" term="Request">
<context id="408730004" term="procedure context" defaultId="385644000" defaultTerm="requested"
permitted="385644000" permittedTerm="requested"/>
<context id="408731000" term="temporal context" defaultId="410512000" defaultTerm="current or specified"
permitted="410510008" permittedTerm="temporal context value"/>
<context id="408732007" term="subject relationship context" defaultId="410604004" defaultTerm="subject
of record" permitted="125676002" permittedTerm="person - add other permitted values in future"/>
</moodCode>
<moodCode code="PRP" term="Proposal">
<context id="408730004" term="procedure context" defaultId="385643006" defaultTerm="to be done"
permitted="385649005 385643006" permittedTerm="being organised / to be done"/>
<context id="408731000" term="temporal context" defaultId="410512000" defaultTerm="current or specified"
permitted="410510008" permittedTerm="temporal context value"/>
<context id="408732007" term="subject relationship context" defaultId="410604004" defaultTerm="subject
of record" permitted="125676002" permittedTerm="person - add other permitted values in future"/>
</moodCode>
<moodCode code="PRMS" term="Promise">
<context id="408730004" term="procedure context" defaultId="385645004" defaultTerm="accepted"
permitted="385649005 385645004" permittedTerm="being organised / accepted"/>
<context id="408731000" term="temporal context" defaultId="410512000" defaultTerm="current or specified"
permitted="410510008" permittedTerm="temporal context value"/>
<context id="408732007" term="subject relationship context" defaultId="410604004" defaultTerm="subject
of record" permitted="125676002" permittedTerm="person - add other permitted values in future"/>
</moodCode>
<moodCode code="ARQ" term="Appointment request">
<context id="408730004" term="procedure context" defaultId="385644000" defaultTerm="requested"
permitted="385644000" permittedTerm="requested"/>
<context id="408731000" term="temporal context" defaultId="410512000" defaultTerm="current or specified"
permitted="410510008" permittedTerm="temporal context value"/>
<context id="408732007" term="subject relationship context" defaultId="410604004" defaultTerm="subject
of record" permitted="125676002" permittedTerm="person - add other permitted values in future"/>
</moodCode>
<moodCode code="APT" term="Appointment">
<context id="408730004" term="procedure context" defaultId="60304008" defaultTerm="scheduled"
permitted="60304008" permittedTerm="scheduled"/>
<context id="408731000" term="temporal context" defaultId="410512000" defaultTerm="current or specified"
permitted="410510008" permittedTerm="temporal context value"/>
<context id="408732007" term="subject relationship context" defaultId="410604004" defaultTerm="subject
of record" permitted="125676002" permittedTerm="person - add other permitted values in future"/>
</moodCode>
<moodCode code="DEF" term="Definition" notApplicable="true"/>
<moodCode code="SLOT" term="Resource slot" notApplicable="true"/>
<moodCode code="EVN.CRT" term="Event criterion" notApplicable="true"/>
<moodCode code="OPT" term="Option" notApplicable="true"/>
</moodMap>

```

Figure 120: MoodMap.xml example

7.8.2.4.4.3.3 Apply context



If no *context wrapper* is provided by the *Resolve context* process, the context-free expression from the *Separate expression context* process is returned as the fully normalized expression.

If the *Resolve context* process provides a *context wrapper* including a 2470590016 | finding context |attribute, the context-free expression from the *Separate expression context* process is applied to this as the value of the 246090004 | associated finding |attribute. The resulting context-dependent expression is returned as the fully normalized expression.

If the *Resolve context* process provides a *context wrapper* including a 2470591017 | procedure context | attribute, the context-free expression from the *Separate expression context* process is applied to this as the value of the 363589002 | associated procedure | attribute. The resulting context-dependent expression is returned as the fully normalized expression.

moodMap																																				
moodCode (13)		code	term	context	context																															
		Any		context (4)																																
1	-			<table border="1"> <thead> <tr> <th>= id</th> <th>= term</th> <th>= defaultId</th> <th>= defaultTerm</th> <th>= permitted</th> <th>= permittedTerm</th> </tr> </thead> <tbody> <tr> <td>1 408729009</td> <td>finding context</td> <td>410515003</td> <td>known present</td> <td>410514004</td> <td>finding context value</td> </tr> <tr> <td>2 408730004</td> <td>procedure context</td> <td>385658003</td> <td>done</td> <td>288532009</td> <td>context values for actions</td> </tr> <tr> <td>3 408731000</td> <td>temporal context</td> <td>410512000</td> <td>current or specified</td> <td>410510008</td> <td>temporal context value</td> </tr> <tr> <td>4 408732007</td> <td>subject relationship context</td> <td>410604004</td> <td>subject of record</td> <td>125676002</td> <td>person - add other permitted values in future</td> </tr> </tbody> </table>	= id	= term	= defaultId	= defaultTerm	= permitted	= permittedTerm	1 408729009	finding context	410515003	known present	410514004	finding context value	2 408730004	procedure context	385658003	done	288532009	context values for actions	3 408731000	temporal context	410512000	current or specified	410510008	temporal context value	4 408732007	subject relationship context	410604004	subject of record	125676002	person - add other permitted values in future		
= id	= term	= defaultId	= defaultTerm	= permitted	= permittedTerm																															
1 408729009	finding context	410515003	known present	410514004	finding context value																															
2 408730004	procedure context	385658003	done	288532009	context values for actions																															
3 408731000	temporal context	410512000	current or specified	410510008	temporal context value																															
4 408732007	subject relationship context	410604004	subject of record	125676002	person - add other permitted values in future																															
2	EVN	Event		<table border="1"> <thead> <tr> <th>= id</th> <th>= term</th> <th>= defaultId</th> <th>= defaultTerm</th> <th>= permitted</th> <th>= permittedTerm</th> </tr> </thead> <tbody> <tr> <td>1 408729009</td> <td>finding context</td> <td>410515003</td> <td>known present</td> <td>410514004</td> <td>finding context value</td> </tr> <tr> <td>2 408730004</td> <td>procedure context</td> <td>385658003</td> <td>done</td> <td>410523001</td> <td>post-starting action status</td> </tr> <tr> <td>3 408731000</td> <td>temporal context</td> <td>410512000</td> <td>current or specified</td> <td>410510008</td> <td>temporal context value</td> </tr> <tr> <td>4 408732007</td> <td>subject relationship context</td> <td>410604004</td> <td>subject of record</td> <td>125676002</td> <td>person - add other permitted values in future</td> </tr> </tbody> </table>	= id	= term	= defaultId	= defaultTerm	= permitted	= permittedTerm	1 408729009	finding context	410515003	known present	410514004	finding context value	2 408730004	procedure context	385658003	done	410523001	post-starting action status	3 408731000	temporal context	410512000	current or specified	410510008	temporal context value	4 408732007	subject relationship context	410604004	subject of record	125676002	person - add other permitted values in future		
= id	= term	= defaultId	= defaultTerm	= permitted	= permittedTerm																															
1 408729009	finding context	410515003	known present	410514004	finding context value																															
2 408730004	procedure context	385658003	done	410523001	post-starting action status																															
3 408731000	temporal context	410512000	current or specified	410510008	temporal context value																															
4 408732007	subject relationship context	410604004	subject of record	125676002	person - add other permitted values in future																															
3	GOL	Goal		<table border="1"> <thead> <tr> <th>= id</th> <th>= term</th> <th>= defaultId</th> <th>= defaultTerm</th> <th>= permitted</th> <th>= permittedTerm</th> </tr> </thead> <tbody> <tr> <td>1 408729009</td> <td>finding context</td> <td>410518001</td> <td>goal</td> <td>410518001</td> <td>goal</td> </tr> <tr> <td>2 408731000</td> <td>temporal context</td> <td>410512000</td> <td>current or specified</td> <td>410510008</td> <td>temporal context value</td> </tr> <tr> <td>3 408732007</td> <td>subject relationship context</td> <td>410604004</td> <td>subject of record</td> <td>125676002</td> <td>person - add other permitted values in future</td> </tr> </tbody> </table>	= id	= term	= defaultId	= defaultTerm	= permitted	= permittedTerm	1 408729009	finding context	410518001	goal	410518001	goal	2 408731000	temporal context	410512000	current or specified	410510008	temporal context value	3 408732007	subject relationship context	410604004	subject of record	125676002	person - add other permitted values in future								
= id	= term	= defaultId	= defaultTerm	= permitted	= permittedTerm																															
1 408729009	finding context	410518001	goal	410518001	goal																															
2 408731000	temporal context	410512000	current or specified	410510008	temporal context value																															
3 408732007	subject relationship context	410604004	subject of record	125676002	person - add other permitted values in future																															
4	INT	Intent		<table border="1"> <thead> <tr> <th>= id</th> <th>= term</th> <th>= defaultId</th> <th>= defaultTerm</th> <th>= permitted</th> <th>= permittedTerm</th> </tr> </thead> <tbody> <tr> <td>1 408730004</td> <td>procedure context</td> <td>410522006</td> <td>pre-starting action status</td> <td>410522006</td> <td>pre-starting action status</td> </tr> <tr> <td>2 408731000</td> <td>temporal context</td> <td>410512000</td> <td>current or specified</td> <td>410510008</td> <td>temporal context value</td> </tr> <tr> <td>3 408732007</td> <td>subject relationship context</td> <td>410604004</td> <td>subject of record</td> <td>125676002</td> <td>person - add other permitted values in future</td> </tr> </tbody> </table>	= id	= term	= defaultId	= defaultTerm	= permitted	= permittedTerm	1 408730004	procedure context	410522006	pre-starting action status	410522006	pre-starting action status	2 408731000	temporal context	410512000	current or specified	410510008	temporal context value	3 408732007	subject relationship context	410604004	subject of record	125676002	person - add other permitted values in future								
= id	= term	= defaultId	= defaultTerm	= permitted	= permittedTerm																															
1 408730004	procedure context	410522006	pre-starting action status	410522006	pre-starting action status																															
2 408731000	temporal context	410512000	current or specified	410510008	temporal context value																															
3 408732007	subject relationship context	410604004	subject of record	125676002	person - add other permitted values in future																															
5	RQO	Request		<table border="1"> <thead> <tr> <th>= id</th> <th>= term</th> <th>= defaultId</th> <th>= defaultTerm</th> <th>= permitted</th> <th>= permittedTerm</th> </tr> </thead> <tbody> <tr> <td>1 408730004</td> <td>procedure context</td> <td>385644000</td> <td>requested</td> <td>385644000</td> <td>requested</td> </tr> <tr> <td>2 408731000</td> <td>temporal context</td> <td>410512000</td> <td>current or specified</td> <td>410510008</td> <td>temporal context value</td> </tr> <tr> <td>3 408732007</td> <td>subject relationship context</td> <td>410604004</td> <td>subject of record</td> <td>125676002</td> <td>person - add other permitted values in future</td> </tr> </tbody> </table>	= id	= term	= defaultId	= defaultTerm	= permitted	= permittedTerm	1 408730004	procedure context	385644000	requested	385644000	requested	2 408731000	temporal context	410512000	current or specified	410510008	temporal context value	3 408732007	subject relationship context	410604004	subject of record	125676002	person - add other permitted values in future								
= id	= term	= defaultId	= defaultTerm	= permitted	= permittedTerm																															
1 408730004	procedure context	385644000	requested	385644000	requested																															
2 408731000	temporal context	410512000	current or specified	410510008	temporal context value																															
3 408732007	subject relationship context	410604004	subject of record	125676002	person - add other permitted values in future																															
6	PRP	Proposal		<table border="1"> <thead> <tr> <th>= id</th> <th>= term</th> <th>= defaultId</th> <th>= defaultTerm</th> <th>= permitted</th> <th>= permittedTerm</th> </tr> </thead> <tbody> <tr> <td>1 408730004</td> <td>procedure context</td> <td>385643006</td> <td>to be done</td> <td>385649005 385643006</td> <td>being organised / to be done</td> </tr> <tr> <td>2 408731000</td> <td>temporal context</td> <td>410512000</td> <td>current or specified</td> <td>410510008</td> <td>temporal context value</td> </tr> <tr> <td>3 408732007</td> <td>subject relationship context</td> <td>410604004</td> <td>subject of record</td> <td>125676002</td> <td>person - add other permitted values in future</td> </tr> </tbody> </table>	= id	= term	= defaultId	= defaultTerm	= permitted	= permittedTerm	1 408730004	procedure context	385643006	to be done	385649005 385643006	being organised / to be done	2 408731000	temporal context	410512000	current or specified	410510008	temporal context value	3 408732007	subject relationship context	410604004	subject of record	125676002	person - add other permitted values in future								
= id	= term	= defaultId	= defaultTerm	= permitted	= permittedTerm																															
1 408730004	procedure context	385643006	to be done	385649005 385643006	being organised / to be done																															
2 408731000	temporal context	410512000	current or specified	410510008	temporal context value																															
3 408732007	subject relationship context	410604004	subject of record	125676002	person - add other permitted values in future																															
7	PRMS	Promise		<table border="1"> <thead> <tr> <th>= id</th> <th>= term</th> <th>= defaultId</th> <th>= defaultTerm</th> <th>= permitted</th> <th>= permittedTerm</th> </tr> </thead> <tbody> <tr> <td>1 408730004</td> <td>procedure context</td> <td>385645004</td> <td>accepted</td> <td>385649005 385645004</td> <td>being organised / accepted</td> </tr> <tr> <td>2 408731000</td> <td>temporal context</td> <td>410512000</td> <td>current or specified</td> <td>410510008</td> <td>temporal context value</td> </tr> <tr> <td>3 408732007</td> <td>subject relationship context</td> <td>410604004</td> <td>subject of record</td> <td>125676002</td> <td>person - add other permitted values in future</td> </tr> </tbody> </table>	= id	= term	= defaultId	= defaultTerm	= permitted	= permittedTerm	1 408730004	procedure context	385645004	accepted	385649005 385645004	being organised / accepted	2 408731000	temporal context	410512000	current or specified	410510008	temporal context value	3 408732007	subject relationship context	410604004	subject of record	125676002	person - add other permitted values in future								
= id	= term	= defaultId	= defaultTerm	= permitted	= permittedTerm																															
1 408730004	procedure context	385645004	accepted	385649005 385645004	being organised / accepted																															
2 408731000	temporal context	410512000	current or specified	410510008	temporal context value																															
3 408732007	subject relationship context	410604004	subject of record	125676002	person - add other permitted values in future																															
8	ARQ	Appointment request		<table border="1"> <thead> <tr> <th>= id</th> <th>= term</th> <th>= defaultId</th> <th>= defaultTerm</th> <th>= permitted</th> <th>= permittedTerm</th> </tr> </thead> <tbody> <tr> <td>1 408730004</td> <td>procedure context</td> <td>385644000</td> <td>requested</td> <td>385644000</td> <td>requested</td> </tr> <tr> <td>2 408731000</td> <td>temporal context</td> <td>410512000</td> <td>current or specified</td> <td>410510008</td> <td>temporal context value</td> </tr> <tr> <td>3 408732007</td> <td>subject relationship context</td> <td>410604004</td> <td>subject of record</td> <td>125676002</td> <td>person - add other permitted values in future</td> </tr> </tbody> </table>	= id	= term	= defaultId	= defaultTerm	= permitted	= permittedTerm	1 408730004	procedure context	385644000	requested	385644000	requested	2 408731000	temporal context	410512000	current or specified	410510008	temporal context value	3 408732007	subject relationship context	410604004	subject of record	125676002	person - add other permitted values in future								
= id	= term	= defaultId	= defaultTerm	= permitted	= permittedTerm																															
1 408730004	procedure context	385644000	requested	385644000	requested																															
2 408731000	temporal context	410512000	current or specified	410510008	temporal context value																															
3 408732007	subject relationship context	410604004	subject of record	125676002	person - add other permitted values in future																															
9	APT	Appointment		<table border="1"> <thead> <tr> <th>= id</th> <th>= term</th> <th>= defaultId</th> <th>= defaultTerm</th> <th>= permitted</th> <th>= permittedTerm</th> </tr> </thead> <tbody> <tr> <td>1 408730004</td> <td>procedure context</td> <td>603044008</td> <td>scheduled</td> <td>603044008</td> <td>scheduled</td> </tr> <tr> <td>2 408731000</td> <td>temporal context</td> <td>410512000</td> <td>current or specified</td> <td>410510008</td> <td>temporal context value</td> </tr> <tr> <td>3 408732007</td> <td>subject relationship context</td> <td>410604004</td> <td>subject of record</td> <td>125676002</td> <td>person - add other permitted values in future</td> </tr> </tbody> </table>	= id	= term	= defaultId	= defaultTerm	= permitted	= permittedTerm	1 408730004	procedure context	603044008	scheduled	603044008	scheduled	2 408731000	temporal context	410512000	current or specified	410510008	temporal context value	3 408732007	subject relationship context	410604004	subject of record	125676002	person - add other permitted values in future								
= id	= term	= defaultId	= defaultTerm	= permitted	= permittedTerm																															
1 408730004	procedure context	603044008	scheduled	603044008	scheduled																															
2 408731000	temporal context	410512000	current or specified	410510008	temporal context value																															
3 408732007	subject relationship context	410604004	subject of record	125676002	person - add other permitted values in future																															
10	DEF	Definition																																		
11	SLOT	Resource slot																																		
12	EVN.CRT	Event criterion																																		
13	OPT	Option																																		

Figure 121: MoodMap.xml file image

7.8.2.4.4.4 Additional steps for alternative forms



The processes described in the preceding sections generate the "long normal view". This is the most general *normal form*. It can be directly applied to meet key requirements related to subsumption testing. It can also be used as the source from which to derive other useful forms that are optimized for particular purposes. The following sections outline some of these.

7.8.2.4.4.4.1 Deriving the short normal view



The short *normal form* can be derived from the long *normal form* by the following steps.

7.8.2.4.4.4.1.1 Generate the set of normalized definitions of each primitive focus concepts



A normalized expression includes the normalized definition for concept in the original expression.

- The normalized definition includes. In the long *normal form* this process is extended to include the definitions of the *Primitive concepts* in the normalized expression.

- The values of all defining *relationship* are expanded and normalized by recursively following the full set of rules described in [Normalize focus concepts](#) on page 368.
-  **Note:** Storage of pre-computed normalized forms of *concept* definitions simplifies this process as it removes the requirement for recursive processing of definitions at run time.

7.8.2.4.4.1.2 Merge the generated definition sets



This process follows exactly the steps described in [Merge definitions](#).

7.8.2.4.4.1.3 Removed redundant attributes and groups



Attributes and groups shared with the merged definition are removed from the *refinement*. Only groups and ungrouped attributes that are identical can be removed from the *refinement*. If a group is not identical the parts that are similar cannot be removed.

7.8.2.4.4.1.4 Recursive removal of redundancy



The process described in this section is recursively applied to any nested *expressions* that remain after the top-level process to remove redundant attributes and groups.

Unlike the process of normalization , this process is done breadth first at each level in the *hierarchy*. If long normalized forms at nested levels are shortened before checking for redundancy, the *expression* will not match those in the merged definition even if they are semantically identical.

7.8.2.4.4.4.2 Canonical representations



The idea of a canonical representation is that it generates a predictable *string* rendering. The missing element to deliver this in the *description* of the "long *normal form*", is a specified sort *order* within the collections elements in an *expression*. A standard sort *order* is not essential for general purpose use but it is very useful to enable fast matching of logically identical *expressions* (which might otherwise be obscured by differences in order that have no semantic relevance).

The *canonical form* for an *expression* is regarded as being the long *normal form* ordered according to the following sorting rules.

- The *expression* is rendered in the form specified by the *SNOMED CT compositional grammar*. For canonical representation a restricted version of the *compositional grammar* is used:
 - No whitespace characters may be included in the *canonical form*
 - No pipe characters "|" and thus no *term* text shall be included in the *canonical form*.
 - Thus the only permitted characters are:
 - Digits [0-9] - for conceptId values;
 - Plus [+] - to combine *focus concepts*;
 - Colon [:] - to represent the start of a *refinement*;
 - Equals [=] - to link an *attribute name* to its value;
 - Comma [,] - to separate attributes within a *refinement*
 - Round brackets [()] - to represent nesting;
 - Curly brackets [{ }] - to represent grouping.
- The syntax determines the general *order* of elements within an *expression* as follows:
 - Focus conceptIds;
 - Attributes* (expressed as name-value pairs);
 - Groups (containing attributes).
- Within a set of focus conceptIds:
 - Concept Identifiers* are sorted alphabetically based on their normal *string* rendering (i.e. digits with no leading zeros):

- The reason for alphabetic sorting rather than numeric sorting is that it is complex to sort attributes and groups which consist of an arbitrary number of conceptIds using numeric keys.
- Within a set of ungrouped attributes or a set of attributes within a group:
 - *Attributes* are sorted alphabetically based on the *string* concatenation of the name and value conceptIds separated by an "=" sign;
 - If a value contains nested *refinements*, the value is enclosed in round brackets (which may influence the sort *order*) and the elements of the nested *expression* are sorted by applying the general canonical sorting rules.
- Within a set of groups:
 - Groups are sorted by alphabetical *order* of the combined set of previously sorted attributes.

7.8.2.4.5 Testing subsumption and equivalence between expressions



The main *reason* for generating *normal form expressions* is to enable testing for *equivalence* and subsumption between different post coordinated *expressions*. This section describes how these processes are carried out.

The process of generating *normal form* for an *expression* also requires testing of subsumption between subsidiary elements within the *expression*.

7.8.2.4.5.1 Testing for equivalence



The following steps can be applied to test for *equivalence* between any two valid *expressions*.

1. Transform both *expression* to long *normal form* (see [Transforming expressions to normal forms](#)).
2. Render these *normal forms* according the canonical representation (see [Canonical representations](#)).
3. Perform a simple *string* comparison between the two long *normal forms* in canonical representation:
 - a. If the *strings* are identical then the *expressions* being tested are equivalent;
 - b. If the *strings* are not identical the two *expressions* being tested are not logically equivalent.

Note that this does not prove that the *expressions* are not equivalent. This limitation applies for the following *reasons*:

- One or more of the *concepts* referenced by the original *expressions* may be *primitive* (i.e. not *fully defined*) and this may obscure the *equivalence*¹⁶.
- Two different *expressions* may include an alternative "sufficient set" of *attributes* that imply the same meaning (see [Nature of the definition](#)).

7.8.2.4.5.2 Testing expression subsumption



The following steps can be applied to test for subsumption of any *candidate expression* by a *predicate expression*.

1. Transform the *predicate expression* to short *normal form*¹⁷(see [Deriving the short normal view](#) on page 379):
 - The resulting "predicate short *normal form expression*" is referred in subsequent steps as the *normalized-predicate*.

¹⁶ This issue will gradually diminish in significance as more concepts are fully defined through addition of new defining relationships.

¹⁷ The *predicate long normal form* can be used instead of the *predicate short normal form*. However, the short form is preferred as it reduces the number of steps required in testing each candidate expression.

2. Transform the candidate expression to long *normal form* (see [Transforming expressions to normal forms](#)) :
 - The resulting "candidate long *normal form expression*" is referred in subsequent steps as the *normalized -candidate*.
3. Test for subsumption between the *normalized -predicate* and the *normalized -candidate* by applying the tests described in [Testing subsumption between two normal form expressions](#) on page 382:
 - The *predicate expression* subsumes the *candidate expression* if the *normalized -predicate* subsumes the *normalized -candidate*.

7.8.2.4.5.2.1 Testing subsumption between two normal form expressions



The following steps are applied to test if a *normalized -predicate* subsumes a *normalized -candidate*. This assumes that these *normal form expressions* have been generated as outlined in [Transforming expressions to normal forms](#).

1. Test that each *focus concept* referenced in the *normalized -predicate* subsumes at least one *focus concept* in the *normalized -candidate*:
 - If not, the *normalized -predicate* does not subsume the *normalized -candidate*. No further testing is required:
 - Exit with result *false*.
 - The approach to testing *concept* subsumption is described in section [Testing concept subsumption](#) on page 384.
2. Test that each *attribute group* in the *normalized -predicate* subsumes at least one *attribute group* in the *normalized -candidate*:
 - If not, the *normalized -predicate* does not subsume the *normalized -candidate*. No further testing is required:
 - Exit with result *false*.
 - The approach to testing *attribute group* subsumption is described in [Testing subsumption between two attribute groups](#) on page 382
3. Test that each ungrouped *attribute* in the *normalized -predicate* subsumes at least one *attribute* (either grouped or ungrouped) in the *normalized -candidate*:
 - If not, the *normalized -predicate* does not subsume the *normalized -candidate*:
 - Exit with result *false*.
 - The approach to testing *attribute* subsumption is described in [Testing attribute subsumption](#) on page 384.
4. If all these tests succeed, the *normalized-predicate* subsumes the *normalized-candidate*:
 - Exit with result *true*.

7.8.2.4.5.2.2 Testing subsumption between two attribute groups



The following steps test if a *predicate -attribute group* subsumes *candidate -attribute group*.

1. Check the *predicate -attribute group* for the presence of the *attribute*: 408729009 | finding context |:
 - If the group does not contain this *attribute*, apply the normal *attribute group* tests specified in [Testing a normal attribute group](#) on page 383.

2. If the *predicate -attribute group* contains the 408729009 | finding context | *attribute*, check whether its value is one of the following: 410516002 | known absent | or 410594000 | definitely not present |:
 - If the *attribute* exists and has one of these values, apply the tests for a *context attribute group with absent finding*, as specified in [Testing a context attribute group with absent finding](#) on page 383;
 - If the *attribute* exists and has any other value, apply the tests for a normal *attribute group*, as specified in [Testing a normal attribute group](#) on page 383.

7.8.2.4.5.2.2.1 Testing a normal attribute group



The following step tests most *attribute groups*. However, a modified approach (see [Testing a context attribute group with absent finding](#) on page 383) is required in the case of *attribute groups* that indicate the absence of a finding.

1. Test that each *attribute* in the *predicate -attribute group* subsumes at least one *attribute* in the *candidate -attribute group*:
 - If not, the *predicate -attribute group* does not subsume the *candidate -attribute group*:
 - Exit with result *false*.
 - The approach to testing *attribute* subsumption is described in [Testing attribute subsumption](#) on page 384
2. If all *attributes* in the group pass this test, the *predicate -attribute group* subsumes the *candidate -attribute group*:
 - Exit with result *true*.

7.8.2.4.5.2.2.2 Testing a context attribute group with absent finding



The following steps test most *attribute groups* that indicate the absence of a finding. This approach differs from the general tests applicable to other *attribute groups* because of the way in which assertions of absence affect the direction of subsumption. This is discussed in detail in [Recording and retrieving absent findings](#).

1. Attempt to match each *attribute* in the *predicate -attribute group* with an *Attribute* which has the same name in the *candidate -attribute group*:
 - If any *attribute* in the *predicate -attribute group* is not matched by an *Attribute* with same name in the *candidate -attribute group*, the *predicate -attribute group* does not subsume the *candidate -attribute group*:
 - Exit with result *false*.
2. For each of the matched *attributes* identified in the previous step, compare the value of the *attribute* in the *predicate -attribute group* with the value of the same *attribute* in the *candidate -attribute group*:
 - If the *attribute name* is 408729009 | finding context | or 408731000 | temporal context |, the *candidate-value* must be equivalent to or subsumed by the *predicate-value*.
 - However, if the *attribute name* is 246090004 | associated finding | or 408732007 | subject relationship context |, the direction of the test is inverted. In these cases, the *predicate-value* must be equivalent to or subsumed by the *candidate-value*.
 - If any of these tests fail, the *predicate -attribute group* does not subsume the *candidate -attribute group*:
 - Exit with result *false*.
 - *Attribute values* are *expressions* and are tested in the same way as any other *expressions* (see [Testing subsumption between two normal form expressions](#)):
 - *Expression* subsumption testing is recursive where *expressions* include nested *qualifiers*.

3. If all the tests above are successful, the *predicate -attribute group* subsumes the *candidate -attribute group*:
 - Exit with result *true*.

7.8.2.4.5.2.3 Testing attribute subsumption



The following steps test if a *predicate -Attribute* subsumes a *candidate -attribute*.

1. Test that the candidate *attribute name* is either the same as or subsumed by the predicate *attribute name*:
 - If not, the *predicate -attribute* does not subsume the *candidate -attribute*:
 - Exit with result *false*.
 - The approach to testing *concept* subsumption is described in [Testing concept subsumption](#).
2. Test that the *candidate -attribute* value is equivalent to or subsumed by the *predicate -attribute* value:
 - If not, the *predicate -attribute* does not subsume the *candidate -attribute*:
 - Exit with result *false*.
 - *Attribute values are expressions* and are tested in the same way as any other expression (see [Testing subsumption between two normal form expressions](#)):
 - Expression subsumption testing is recursive where *expressions* include nested *qualifiers*.
3. If both the above tests are successful, the *predicate -attribute* subsumes the *candidate -attribute*:
 - Exit with result *true*.

7.8.2.4.5.2.4 Testing concept subsumption



The following steps test if a *predicate - concept* subsumes a *candidate - concept*.

1. Test if candidate - *concept* is an *inactive concept*:
 - *candidate - concept.conceptStatus NOT IN (0, 6, 11)*:
 - If the *candidate - concept* is *inactive* then look for an *active concept* related by a *historical relationship* | SAME AS | or | REPLACED BY | and treat this as the *candidate - concept* in subsequent steps¹⁸.
2. Test if the *candidate - concept* is identical to the *predicate - concept* :
 - If *candidate - concept.conceptId == predicate - concept.conceptId* the *concepts* are identical:
 - Exit with result *true* (accept equivalent).
3. Test if the *predicate - concept* is one of the *supertype ancestors* of the *candidate - concept*.
 - This is true if a sequence of | is a | relationships leads from the *candidate - concept* (as source - *conceptId1*) to the *predicate - concept* (as the target - *conceptId2*):
 - Exit returning the result of this test.
 - Various approaches to optimization of this test are described in [Optimizing concept subsumption testing](#). The recommended approach is to use a *transitive closure* table (see [Transitive Closure Implementation](#)).

¹⁸ Optionally active concepts that are related to ambiguous candidate -concepts by | MAY BE A | historical relationships could also be tested. However, this requires a decision as to whether the prime objective of retrieval is "completeness" (in which case include these possible related concepts) or "precision" in which case they should be excluded.

7.8.2.4.6 Optimization of normalization and expression subsumption testing



The steps in the normal *transform* and subsumption testing processes are not particularly onerous. However, queries require thousands or millions of such tests to be carried out. It is therefore likely that most practical implementations will require some type of optimization to support tests for subsumption between *expressions*.

The method described in this section is one approach to optimization . The central idea is the use of a repository to store *expressions* and *relationships* between *expressions*.

The advantages of this include the following:

- All transformcomputations can be done off-line rather than at run-time;
- Less *transforms* are done as each distinct candidate *expression* need only be transformed once when created and once more each time a new release alters the definitions on which it is based:
 - Other approaches either require:
 - real - time *transformation* each time a candidate *expression* is considered for retrieval;
 - or
 - storage of *normal forms* in each record entry and updating of each *normal form* instance whenever a new release affects the definitions on which it is based.

Neither of these approaches appears to be scalable over time, as record volumes increase. In contrast the proposed optimization is not affected by the total number of records but only by the total number of distinct *expressions* encountered.

- Additional optimization is possible by pre-classifying the repository so that individual queries can test an *expression* with a single join to a table representing the *transitive closure* of all used *expressions*.

The approach described in the following section is only one way of implementing the central idea of optimization using an *expression* repository. There several ways to harness the same general technique and some of these may be better suited to particular requirements or technical environments.

7.8.2.4.6.1 Expression repository design



The primary requirements for an *expression* repository are:

- Allocation of a fixed length, unique *Identifier* for every *expression* used in an operational environment:
 - The size of an operational environment may range from an individual application at a particular site to a large multi-site organization using multiple applications that use the same *expression* repository.
 - The easiest way to deliver unique *Identifiers* within this range of organizational scales is the use of a *UUID* (also referred to a *GUID*). The *UUID/GUID* allocation algorithm provides an industry standard approach to allocation of universally unique 128-bit *Identifiers* and is readily available in all widely used operating systems.
- Linking every close-to-user *expression* with its long *normal form*:
 - It is necessary to update this link each time a new release of *SNOMED CT* changes an underlying *relationship* that may affect the *normal form*.
- The *expressions* themselves need to be searchable:
 - The canonical version of the *SNOMED CT compositional grammar* is recommended for this purpose because it has a minimum of syntactic noise and a specified sort *order* for the elements within the *expression*.
 - Despite these advantages some *normal form expressions* can be quite long. Currently the longest *normal forms* seen are up to 300 characters long. For a degree of future proofing it is suggested that the longest indexable variable length character *string* should be used (e.g. in MS SQL Server a length of up to 900 characters).

Two possible designs that meet this goal are suggested in next two sections.

- The first option uses two tables - one to identify *expressions* and the other to link them to indicate the results of *transformation* (see [Dual table expression repository](#)).
- The second approach is less flexible and in its *current* form it lacks many of the features of the first option. It is included in this guide because it follows an original suggested design and indicates an alternative for consideration and discussion. (see [Single table expression repository](#)).

Whichever of these designs is used the general steps in using the tables is similar (see [Using the expressions repository](#))

7.8.2.4.6.1.1 Dual table expression repository



A dual table design is more flexible and potentially more compact than a [single table expression repository](#). The compactness is achieved because each distinct *normal form* only occurs once in one row of the table. The flexibility results from the ability to make multiple links between *expressions* to specify the results of different *transforms* without repeating the *expression*.

Each discrete *expression* (whether close-to-user or one of the *normal forms*) is allocated a unique ExpressionId when it is first used or generated. From this point on, this ExpressionId is immutably linked to the *expression* and the *expression* must not be altered in anyway.

The ExpressionLink table represents the linkage between an *expression* and its *normal forms*. ExpressionLinks can be updated as necessary (i.e. when a new release is received) without any effect on the existing content of the *Expression* table. However, an additional row will be added to the *expression* table whenever a *transform* results in a new *expression* (i.e. any *expression* that is not in the *Expression* table).

Table 259: Suggested structure for the Expression table in a dual table expression repository

<p>Each row in the <i>Expression</i> table represents and identifies an <i>expression</i>. All <i>expressions</i> used are identified in this way (i.e. close-to-user and <i>normal forms</i>) and the links between an <i>expression</i> and a transformed version of that <i>expression</i> are specified by the <i>ExpressionLink</i> table.</p>				
Primary Key	Field Type	Permitted characters	Length	Description
ExpressionId	GUID	binary-or - string version	16/36	Unique Identifier for this expression.
Data Fields	Field Type	Permitted characters	Length	Description
Expression	String	0 to 9 and {} () - , + = :	6 -900	Canonical rendering of an expression in SNOMED CT compositional grammar *Indexed*
DateAdded	IsoDateTime	binary-or - string version	8/20	Date time of addition of this Expression to the expressions table. YYYYMMDDhhmmss+ZZ.zzz

Table 260: Suggested structure for the ExpressionLink table in a dual table expression repository

<p>Each row in the ExpressionLink table links a source <i>Expression</i> with the result of transforming that <i>expression</i>. The DateIn and DateOut columns allow active and <i>inactive</i> links to be stored in the same table - easing historical review of changes. The primary key ExpressionLinkId allows multiple rows to link the same pair of <i>expressions</i> where a link is valid in one release, not valid in the next and restore in a subsequent release.</p>	<table border="1"> <thead> <tr> <th>Primary Key</th><th>Field Type</th><th>Permitted characters</th><th>Length</th><th>Description</th></tr> </thead> <tbody> <tr> <td>ExpressionLinkId</td><td>GUID</td><td>binary-or - string version</td><td>16/36</td><td></td></tr> </tbody> </table> <p>Unique identifier for this expression link</p>	Primary Key	Field Type	Permitted characters	Length	Description	ExpressionLinkId	GUID	binary-or - string version	16/36		<table border="1"> <thead> <tr> <th>Data Fields</th><th>Field Type</th><th>Permitted characters</th><th>Length</th><th>Description</th></tr> </thead> <tbody> <tr> <td>SourceExpressionId</td><td>GUID</td><td>binary-or - string version</td><td>16/36</td><td>Foreign key link to the <i>Expression</i> table row for the source <i>expression</i> which is linked to a <i>transform</i> by this link. *Indexed*</td></tr> <tr> <td>ResultExpressionId</td><td>GUID</td><td>binary-or - string version</td><td>16/36</td><td>Foreign key link to the <i>Expression</i> table row for an <i>expression</i> representing the result of the <i>transform</i> applied to a the source <i>expression</i> *Indexed*</td></tr> <tr> <td>TransformType</td><td>Enum</td><td>digits [0-9]</td><td>2</td><td>An enumerated value representing the nature of the <i>transform</i> between the source and result <i>expressions</i>. Values might include: 0=Single concept expression Long normal form 1=Other expression Long normal form 2=Long normal form Short normal form A direct <i>transform</i> for each <i>expression</i> to short <i>normal form</i> could be added but is not essential this can be achieved by traversing a type 0 or 1 link followed by a type 2 link.</td></tr> </tbody> </table>	Data Fields	Field Type	Permitted characters	Length	Description	SourceExpressionId	GUID	binary-or - string version	16/36	Foreign key link to the <i>Expression</i> table row for the source <i>expression</i> which is linked to a <i>transform</i> by this link. *Indexed*	ResultExpressionId	GUID	binary-or - string version	16/36	Foreign key link to the <i>Expression</i> table row for an <i>expression</i> representing the result of the <i>transform</i> applied to a the source <i>expression</i> *Indexed*	TransformType	Enum	digits [0-9]	2	An enumerated value representing the nature of the <i>transform</i> between the source and result <i>expressions</i> . Values might include: 0=Single concept expression Long normal form 1=Other expression Long normal form 2=Long normal form Short normal form A direct <i>transform</i> for each <i>expression</i> to short <i>normal form</i> could be added but is not essential this can be achieved by traversing a type 0 or 1 link followed by a type 2 link.
Primary Key	Field Type	Permitted characters	Length	Description																												
ExpressionLinkId	GUID	binary-or - string version	16/36																													
Data Fields	Field Type	Permitted characters	Length	Description																												
SourceExpressionId	GUID	binary-or - string version	16/36	Foreign key link to the <i>Expression</i> table row for the source <i>expression</i> which is linked to a <i>transform</i> by this link. *Indexed*																												
ResultExpressionId	GUID	binary-or - string version	16/36	Foreign key link to the <i>Expression</i> table row for an <i>expression</i> representing the result of the <i>transform</i> applied to a the source <i>expression</i> *Indexed*																												
TransformType	Enum	digits [0-9]	2	An enumerated value representing the nature of the <i>transform</i> between the source and result <i>expressions</i> . Values might include: 0=Single concept expression Long normal form 1=Other expression Long normal form 2=Long normal form Short normal form A direct <i>transform</i> for each <i>expression</i> to short <i>normal form</i> could be added but is not essential this can be achieved by traversing a type 0 or 1 link followed by a type 2 link.																												

Data Fields	Field Type	Permitted characters	Length	Description
Dateln	<i>IsoDateTime</i>	binary-or - string version	8/20	Date time of addition of this CtuExpression to the <i>expressions</i> table. YYYYMMDDhhmmss+ZZ.zz
DateOut	<i>IsoDateTime</i>	binary-or - string version	8/20	Date time at which this ExpressionLink was rendered obsolete by replacement. YYYYMMDDhhmmss+ZZ.zz

7.8.2.4.6.1.2 Single table expression repository



The single table approach provides direct mapping between a close to user *expression* and a *normal form*. This was the original suggested design for an *expressions* table. However, a dual table approach provides a more efficient and more flexible solution (see [Dual table expression repository](#)).

Table 261: Suggested structure for a single table expression repository

CtuExpression Table				
Each row in the <i>Expression</i> table represents a close-to-user form <i>expression</i> and its relationship to a <i>normal form expression</i> .				
Primary Key	Field Type	Permitted characters	Length	Description
ExpressionId	GUID	0 to 9 and A to F and {} -	16/38	Unique Identifier for this close-to-user expression.
Data Key				
CtuExpression	Field Type	Permitted characters	Length	Description
	String	0 to 9 and {} () , + = :	6 - 300	Canonical rendering of close to user <i>expression</i> in SNOMED CT compositional/ grammar *Indexed*
LongNormalExpression	String	0 to 9 and {} () , + = :	6 - 900	Canonical rendering of long <i>normal form expression</i> in SNOMED CT compositional/ grammar *Indexed*
DateUpdated	ISODateTime	0 to 9	20	Date time of last update to the LongNormalExpression for this CtuExpression. YYYYMMDDhhmmss+ZZ..zz
DateAdded	ISODateTime	0 to 9	20	Date time of addition of this CtuExpression to the <i>expressions</i> table. YYYYMMDDhhmmss+ZZ..zz

7.8.2.4.6.2 Using the expressions repository



Whichever approach is taken to the design of the repository the way in which it is used is similar.

7.8.2.4.6.2.1 Run-time data entry and inbound communications



Each time an *expression* is recorded (either directly or in an inbound communication) the *expression* is looked up in the repository. The *expression* is rendered using the canonical version of the *SNOMED CT compositional grammar* and an *expression* matching this *string* is looked for in the repository.

If the *expression* is not found a new row is added to the *expression* repository.

Whether a row is found or added the unique *Identifier* of the *expression* is added to the record entry or other resource in which the information encoded by the *expression* is to be stored. Depending on authentication and other requirements, the original form of the *expression* may also be stored.

This step is often referred to a "just-in - time *precoordination*" because a *precoordinated Identifier* for the *postcoordinated expression* is generated at the time it is required. It is also possible to prime the repository with a range of *expressions* that are anticipated (e.g. because they are generated by a particular set of forms or protocols).

7.8.2.4.6.2.2 Run-time display or outbound communications



Although an *expression* repository may be shared across a large multi-site organization , there are advantages in requiring communication to adhere to standards that are not limited to bounds of that organization and which are not dependent on real - time communication with the repository. Therefore, when there is a requirement to display or communicate the information represented by an *expression Identifier*, the *expression* should be looked up in the repository and added to a communication in its original form.

7.8.2.4.6.2.3 Support for normal form transformation of new expressions



The *transforms* described in this guide are applied to all new *expressions* in the repository.

Where a *transforms* results in a new *expression*, this *expression* is added to the repository. The appropriate reference between the original *expression* and *normal form expression* is created, in a manner determined by the repository design.

7.8.2.4.6.2.4 Support for normal form transformation after updates to SNOMED CT definitions



After a *SNOMED CT* update, the repository is refreshed to check for consequent changes in the *normal forms* for existing *expression*. Where changes are required the appropriate rows in the repository are added or updated in accordance with the repository design.

7.8.2.4.6.2.5 Supporting retrieval with an expression repository - basic



Retrieval requests can be dealt with by using the repository to locate the appropriate *normal forms* for the predicate *expression* and for candidate *expressions*. Instead of requiring processing to *transform expressions* in real-time a simple SQL *query* can immediately return the appropriate *expression*.

The general process of process testing subsumption between *expression* is then applied as documented in this guide (see [Testing subsumption and equivalence between expressions](#)).

7.8.2.4.6.2.6 Supporting retrieval with an expression repository - advanced



Further optimization is possible if the *expressions* in the repository are classified to generate an extended *transitive closure* table. This process is similar in effect to testing for subsumption between every pair of *expressions* in the repository and recording the results of each successful test as a row in a table that identifies the *relationship* between the subsuming and subsumed *expression*. This process may appear to be unscalable because it requires many millions of tests to be carried out. However, fortunately algorithms are available that can optimize this process and classify hundreds of thousands of *concepts* in a little over an hour on what would today be considered a fairly modest system.

If this approach is followed, any predicate *expression* can be tested against any candidate *expression* by a simple SQL *query* using this extended *transitive closure* table. The result of this is that testing subsumption of an *expression* will perform practically as fast as testing the subsumption between two *precoordinated concepts*.

7.8.2.4.6.2.7 Is storing an expression the same as creating a new concept?



In one sense adding an *expression* to a repository and giving it an *Identifier* is the same as creating a new *concept*. However, there are some subtle but highly significant differences between storing, identifying and reusing an *expression* in the way suggested in the guide and a *SNOMED CT concept*. These are summarized in [Differences between concepts and stored expressions](#).

Table 262: Differences between concepts and stored expressions

<i>SNOMED CT released concept</i>	<i>Stored expression</i>
The defining <i>relationships</i> of a <i>SNOMED CT concept</i> are intended to represent the meaning of words or phrases as they are used in clinical practice.	An <i>expression</i> is the collection of references to a set of <i>SNOMED CT concepts</i> .
The meaning of a <i>SNOMED CT concept</i> is represented by the <i>Fully Specified Name</i> (and sometimes by an associated textual definition).	An <i>expression</i> is not associated with a specific <i>text string</i> . It may be rendered in different human readable forms but its only source of meaning is the meaning of the <i>concepts</i> it references.
Because a <i>concept</i> definition attempts to express the human understood meaning of a word or phrase the logical definition expressed by its defining <i>relationship</i> may not be sufficient to fully define the <i>concept</i> . In these cases the <i>concept</i> definition is marked as " <i>primitive</i> ".	Because an <i>expression</i> has no specific <i>term</i> or source of meaning other than the <i>focus concept</i> and <i>Attribute</i> it is inherently " <i>fully defined</i> " in that those <i>Attributes</i> fully define what the <i>expression</i> may be used to represent.
A <i>SNOMED CT Concept</i> can be bound to various <i>terms</i> that are deemed to be <i>synonyms</i> in a given <i>language</i> . The <i>SNOMED CT</i> design provides a framework for managing these bindings, correcting errors, supporting translations and tracking the history of changes.	It may be tempting to associate particular words or phrases with an <i>expression</i> . This will inevitably occur in instances in individual record entries. However, <i>terms</i> should not be bound to <i>expressions</i> in a way that suggests a formal persisting association between that <i>term</i> and the class represented by the <i>expression</i> . If such a binding is required a <i>SNOMED CT concept</i> should be requested (or created in an <i>extension</i>) to provide a proper framework for managing that binding.

7.8.2.4.7 Retrieving absent findings



This part of the guide is based on a white paper produced in May 2006 to consider the impact of "negatives" on *transformation*, normalization and *subtype* testing rules. The outcome of this was revision of the rules on subsumption testing in relation to context *attribute groups* that include finding context values indicating that a finding is known to be absent.

7.8.2.4.7.1 Rationale



The wider issue of different types of negation cannot be completely resolved in a short space of time - as has been demonstrated during numerous previous discussions. However, there are some aspects of *current* advice on computation of subsumption that appear to be misleading in relation to *concepts* that express the absence of a finding.

For example, *current subtype* testing rules on the following *expression*:

373572006 | clinical finding absent | :

246090004 | associated finding | = (125605004 | fracture of bone | :

363698007 | finding site | = 71341001 | bone structure of femur |)

normalizes as follows

243796009 | context-dependent category | :
 {246090004 | associated finding | =
 (64572001 | disease | :
 {116676008 | associated morphology | = 72704001 | fracture |
 ,363698007 | finding site | = 71341001 | bone structure of femur | })
 ,408729009 | finding context | = 410516002 | known absent |
 ,408731000 | temporal context | = 410512000 | current or specified |
 ,408732007 | subject relationship context | = 410604004 |subject of record| }

The result of applying "normal" subsumption testing rules is that | no fracture of femur | is subsumed by | no fracture of bone |. Superficially this may seem reasonable, but it will incorrectly cause the inference that a person with a record of | no fracture of femur | has | no fracture of a bone |. This is true if | no fracture of a bone | meant one bone that it not fractured, but the generally understood meaning would be that the patient had no fractured bones.

Thus the objective was to revise the *transformation* and/or *subtype* testing rules to appropriately handle *expressions* that represent absent findings.

7.8.2.4.7.2 Overview



The approach specified in this guide deals with the computational issue of *subtype* testing based in the *current concept model*, *classifier* logic and distribution format of *SNOMED CT*. The approach has been tested and produces reasonable results with *current* data. It also works appropriate with combined presence and absence finding (e.g. "head injury without skull fracture") provided these are modeled using separate context *attribute groups*.

The positive statement in the previous paragraph must be tempered by the knowledge that a logical technical approach is only a part of the solution. Human factors are an important issue when considering the proper processing of *concepts* of absence and other forms of negation. Therefore, it is also necessary to consider what people may mean when they explicitly state the "absence of a finding"; and what other people may intend when they query records to determine the presence or absence of a finding.

The technical approach suggested for subsumption testing *expressions* that involve absence of a finding are valuable only if applied appropriately. Human interpretation may be required to determine the clinical relevance of the results of absent *subtype* tests for a particular purpose.

7.8.2.4.7.3 Testing subsumption of absence of a finding



7.8.2.4.7.3.1 Initial assumptions



The general rules for computation of subsumption of *expressions* and *transformation* to *normal forms* are stated in detail in the *SNOMED CT* document on *transformation to normal forms*. They can be summarized as follows:

When two *expressions* are tested for subsumption, tests are performed recursively on the following elements within the *normal form* of those *expressions*:

- Groups of *attribute value* pairs;
- *Attribute value* pairs;
- Nested *expressions* use to represent values within an *attribute value* pair.

The *normal form* of an *expression* is derived by a set of rules which retain the full semantic meaning of the original *expression* while *transformation* it to a form in which:

- Every referenced *focus concept* is a *primitive concept*
- Every *attribute value* is a normalized *expression*
- Grouping and nesting of *Attributes* is aligned with the *concept model*

- Default context or context derived from the information model is made explicit using SNOMED CT context *Attributes*

7.8.2.4.7.3.2 Identifying expressions that include absence



The *normal form* of any *expression* that represents absence of finding includes the following standard context *Attributes*:

243796009 | situation with explicit context |:

408729009 | finding context | = 410516002 | known absent | (or a subtype)

, 408731000 | temporal context | = <temporal context value>

, 408732007 | subject relationship context | = <subject Relationship context value>

, 246090004 | associated finding | = <a clinical finding or event>

When the value of | finding context | is | known absent | (or one of its *subtypes*) then it may be appropriate to apply *subtype* testing rules based on absence. However, as discussed in [recording and retrieving absent findings](#), the way in which rules are applied depends on the intended results of the *query* and rationale behind recording a negative finding.

7.8.2.4.7.3.3 Testing groups rather than expressions



The relevant information in an *expression* can be regarded as a group of *attributes* as follows:

{ 408729009 | finding context | = 410516002 | known absent | (or a subtype)

, 408731000 | temporal context | = <temporal context value>

, 408732007 | subject relationship context | =

, 246090004 | associated finding | = }

Considering absence at the group level, rather than at the *expression* level, allows account to be taken of *expressions* that refer to presence of one finding and absence of another.

The following style of *expression* represents the presence of "first clinical finding" and the absence of "second clinical finding".

243796009 | situation with explicit context | :

{ 408729009 | finding context | = 410515003 | known present | (or a subtype)

, 408731000 | temporal context | = <temporal context value>

, 408732007 | subject relationship context | =

, 246090004 | associated finding | = }

{ 408729009 | finding context | = 410516002 | known absent | (or a subtype)

, 408731000 | temporal context | = <temporal context value>

, 408732007 | subject relationship context | =

, 246090004 | associated finding | = }

In this case, the first group is tested according to the general subsumption testing rules and the approach to absence may be appropriate to the second group (i.e. the group that includes | finding context | = | known absent |).

The overall *expression*, containing both these groups, is then tested in the general way according to whether the two groups separately pass the relevant test. The general subsumption testing rules allow groups not present in the predicate *expression* to be present in the candidate *expression*. Therefore both of the following predicate *expressions* subsume the candidate *expression* above irrespective of the special rules for handling absence.

Predicate 1 - "first clinical finding present"

243796009 | situation with explicit context | :

{ 408729009 | finding context | = 410515003 | known present | (or a subtype)
 , 408731000 | temporal context | = <temporal context value>
 , 408732007 | subject relationship context | =
 , 246090004 | associated finding | = }

Predicate 2 - "second clinical finding absent"

243796009 | situation with explicit context | :

{ 408729009 | finding context | = 410516002 | known absent | (or a subtype)
 , 408731000 | temporal context | = <temporal context value>
 , 408732007 | subject relationship context | =
 , 246090004 | associated finding | = }

7.8.2.4.7.3.4 Testing | associated finding | in groups containing | known absent |   

If a group contains | finding context | = | known absent | then the test applied to the value of the | associated finding | *Attribute* is changed.

The general purpose test for the value of an *Attribute* is:

- "is the candidate value identical to or a subtype of the predicate value".

The alternative test when the group contains | known absent | is:

- "is the **predicate** value identical to or a subtype of the **candidate** value".

7.8.2.4.7.3.5 Testing | subject relationship context | in groups containing | known absent |   

If a group contains | finding context | = | known absent | then the test applied to the value of the 408732007 | subject relationship context | *Attribute* should also be changed to the alternative form.

- "is the **predicate** value identical to or a subtype of the **candidate** value".

Thus

- | family history of heart disease in father | implies | FH: Cardiac disorder |; but;
- "no family history of heart disease" implies "no family history of heart disease in father".

7.8.2.4.7.3.6 Testing | temporal context | in groups containing | known absent |   

If a group contains | finding context | = | known absent | then the test applied to the value of the | temporal context | *attribute* needs to be carefully considered depending on the intended result of the *query*.

In some cases it may also be changed to the alternative form.

- "is the **predicate** value identical to or a subtype of the **candidate** value".

Thus:

- "currently has asthma" implies "has, or at some time in past had, asthma"; but;
- "did not have headache recently" does not imply "did not have headache in the past".

However, since the value "all times past" is specified for expressing concepts like | never had a headache | the standard *subsumption* test rules may work in some cases.

Since the time aspect in the record is relative to the time of recording while the intended result of a *query* may be relative to a specified time (or the time of the *query*) the use of temporal context in queries requires careful consideration on a *query by query* basis.

7.8.2.4.7.3.7 Differences between subject Relationship and temporal context



The difference between the handling of "408732007 | subject relationship context |" and | temporal context | noted in [Testing |subject relationship context| in groups containing |known absent|](#) and [Testing |temporal context| in groups containing |known absent|](#) may result from a significant difference in value hierarchies.

Thus "no family history of asthma" literally means something like:

"As far as is known, at all times in the past, the disorder asthma was absent from, all members of the subjects family"

The temporal context value *hierarchy* includes the value "all times past" to capture one part of this. However, for the "all Members of the subject's family" we use the same | Person in the family |concept as is used for asserting "at least one Member of the family".

An argument can be made for aligning the approach in both these hierarchies in one of two ways:

1. Removing the value "all times past" from | temporal context | and using "current or past" in its place. Then the *subtype* testing of temporal context would invert in the same way as for the other *Attributes* (i.e. in absence mode the "current or past" would imply all other temporal context ... aka "all times past").
2. Adding "all Members of family" to the subject *relationship* value *hierarchy* and carefully applying this in all negation expressions. In this case, the alternative *subtype* testing would only apply to | associated finding |.

While approach (b) may appear more rational it does seem to have two disadvantages:

- It requires more disciplined use in modeling and in *postcoordination*
- Several new "all" values would be needed - "all Members of paternal family", "all male Members of family", "all known contacts", etc. to allow negatives to be expressed clearly.

7.8.2.4.7.3.8 Impact of nesting context expressions



Currently, the *concept model* does not allow a *subtype* of | Situation with explicit context | to be the value of a defining *Relationship*. However, some potential use cases have been advanced for allowing a | Finding with explicit context | to be the value of an *attribute*. If this is permitted then inclusion of | known absent | in such a nested *expression* would create additional complexity when trying to resolve queries.

Applying the current testing rules at appropriate nested levels may have the desired result. However, there would be an increased risk of double-negatives and similar logical problems. Therefore, until there are real cases to test, the possibility of new exceptions arising cannot be ruled out.

7.8.2.4.7.4 Human factors and testing absence



The reasons for recording information about absent findings and the rationale for attempting to retrieve information about absent findings often differ from and interact with the strict logical interpretation of negation. Specific aspect of this general point are illustrated by the following subsections.

7.8.2.4.7.4.1 Subtype classification of absent findings



When considering subsumption testing as part of the process of classifying the *concepts* in *SNOMED CT* the underlying assumptions is that the comparison process is potentially symmetrical. Thus any two *concepts* can be compared to ask the following questions:

- Are A and B identical? ... if not then
- Is A a *subtype* of B? ... if not then
- Is B a *subtype* of A?

If not then we might possibly be interested in the semantic proximity of the *concepts* for example ...

- What supertypes do A and B share?
- Are there any *concepts* that are subsumed by both A and B.

In this relatively abstract environment it is possible to discuss ideas about | known absent | or | not done |. These ideas may seem theoretically sound while being less readily applicable in practical clinical applications. In some cases the practical view may be more complex than the abstract view but in the case of "absence" it seems possible that considering real use cases may in some ways simplify or at least assist in prioritization.

7.8.2.4.7.4.2 Querying records for absence findings



Subsumption testing in a clinical application is typically concerned with testing instances of *expressions* in clinical records ("candidate *expressions*") against sets of criteria some of which are represented as *expressions* ("predicate *expressions*").

- A predicate is an *expression* against which other *expressions* are tested. Predicate *expression* may be constructed for specific queries or may be developed as reusable part of clinical protocols, decision support rules or report specifications. In these cases, the author of a predicate is someone trying to find out something by querying a record or set of records.
- The candidate is an *expression* that is tested to see if it is subsumed by the predicate. Candidate *expressions* may be constructed directly by the author of a clinical statement (i.e. an instance of an entry in the record) or by an application designer determining the way in which particular user decisions are recorded. Thus the direct or indirect author of the candidate is typically someone wishing to record (or enable the recording of) a finding or procedure in a record. Although the candidate *expression* is a crucial part of subsumption testing its *reason* for existing is not determined by the requirements of a specific *query* but rather by what the user wishes to record.

The more abstract subsumption testing for classification described in the previous section is a prerequisite for effective subsumption testing in clinical applications. However, the differences between the motivations of those constructing predicate and candidate *expressions* mean that subsumption testing in clinical applications is rarely a symmetrical comparison. The typical test is "does this candidate satisfy the criteria?" or in some cases "could this candidate possibly satisfy the criteria?"

When considering absence or other kinds of negation the difference between the perception of the author of an instance of clinical information and the view of the person constructing a *query* may be even more significant. Thus technical rules for testing subsumption of | known absent | finding are only one part of the picture.

To avoid misunderstanding and consequent errors it is worth considering two general questions:

- What are the possible motivations for recording a | known absent | finding?
- What are the possible motivations for specifying retrieval queries for absent findings?

The next two sections identify several different answers to these questions.

7.8.2.4.7.4.3 Motivations for recording a | known absent | finding?



There thousands of possible findings that might be made at every encounter (and theoretically every second). The vast majority of absent finding are not recorded but there are clearly some good reasons for explicitly recording the absence of some findings. These might include:

1. To record that the author asked a question and got a negative response.

Example:

"Family history - No family history of asthma".

Implied meaning - "I asked the patient if anyone in their family has or had asthma and they said 'no'".

2. To record that the author examined/investigated and did not find this.

Example:

"No heart murmur".

Implied meaning - "I listened for a heart murmur and did not hear one".

3. To record a possible conclusion that the author considered and rejected.

 **Example:**

"No meningitis" (as part of an assessment of a patient with a fever and headache).

Implied meaning - "I considered the possibility of meningitis and rejected it".

4. To refute a statement made by someone else.

 **Example:**

"Not appendicitis" (in a record that contains an earlier assertion of "diagnosis appendicitis").

Implied meaning - "The admitting doctor's diagnosis of appendicitis seems to be incorrect".

5. To indicate a change in an earlier assessment.

 **Example:**

| Carcinoma of bronchus excluded | (as part of record in which the same author previously thought this a likely diagnosis).

Implied meaning - "I thought they might have Ca bronchus but following investigations I have now rejected this diagnosis".

6. To note the resolution of finding that was previously present.

 **Example:**

"No abdominal pain" (in a record which has a previous finding of "abdominal pain present").

Implied meaning - "The abdominal pain present on admission has now resolved".

7. To indicated that a finding that is commonly present in associated with another finding is not present in this case.

 **Example:**

"No loss of consciousness" (in the record of patient who has had a head injury).

Implied meaning - "They did not lose consciousness following an injury which potentially could have caused this".

In (1), (2) and (3) the dominant motive may be to assert what was done or considered. However, recording absent findings may also be a part of the process the author followed to organize her thoughts.

Both (4) and (5) record difference in view related to some previous assertion. Where this is the intention a strong case can be made for linking the statements in the record structure. However, this is a possible motivation even if such links are supported by the system or have not been added to this instance.

In the case of (6) the use of absent indicates a change in condition of the patient rather than an update of the diagnosis or interpretation by the clinician.

In case (7) an absent finding is recorded to refine the nature of a specific condition.

There is considerable overlap between these reasons motivations for recording absence. However, the overall motivation for recording an absent finding may or may not be aligned with the rationale for requesting retrieval of negative findings. This mismatch is likely to lead to anomalous results if the assumptions based only on a logical interpretation of negation.

7.8.2.4.7.4.4 Motivations for specifying retrieval of "known absent" finding?



When querying for absence of a finding the most likely motivation is to establish the absence of a finding. In the absence of evidence to the contrary the normal assumption is that an abnormal finding is absent. Furthermore, in most cases a point in time assertion of absence does not imply the finding was never true, nor that could not be true at a future point in time. Thus in most cases, a query for absence is more

concerned with checking that there is no statement indicating the presence of the finding rather than searching for a statement of presence.

There are exceptions to this:

- If the abnormal finding is usually found in association with a confirmed finding:
 - E.g. "absence of chest pain" in a patient with a confirmed "myocardial infarction").
- If the abnormal finding is obscure and may easily have been overlooked:
 - E.g. "Koplik's spots".
- If an assertion of presence of a finding was made by an informer of unknown reliability:
 - E.g. Bystander asserts that patient had a "heart attack" but clinical assessment excludes this.

If these exceptions apply the presence of a statement of an absent finding may be of interest. However, this depends on specific thinking around the question being posed so that the *query* criteria achieve the desired result. It is not enough to simply search for a specific absence and its *subtypes*. The assumptions about presence or absence of a finding must be considered.

Example: determine the number of road accident victims who have been admitted to hospital but have no fractured bones. In practical *terms* the best approach would just be to exclude those with known presence of fracture. The assumption is that, unless a fracture is mentioned, they are not known to have a fracture.

Another possible motivation for looking for absence findings is to monitor or audit the delivery of care and check that appropriate questions have been asked, tests done, possibilities considered, etc. In these cases, the *query* needs to search for both presence and absence ... or possibly for a procedure code representing the appropriate examination or investigation.

Example: Were all patients admitted for routine surgery asked if they had any allergies.

7.8.2.4.7.5 Conclusions on absent findings



There are rational *reasons* for wishing to record and retrieve information about absent findings.

However, there is not a direct one-to-one *relationship* between the motivations for recording absence and the motivations for retrieval.

The suggested technical advice on subsumption testing of known absent findings addresses the logical question of subsumption, but this is only one part of the picture. The meaning implied by recording a known absent finding needs to be considered in the context of the intention of a *query*. When this is understood the alternative *subsumption test* can be applied appropriately to support complete and accurate retrieval.

7.8.2.4.7.6 Procedures / not done /



The use of the | procedure context | value | not done | (and *subtypes*) has similarities to the | finding context | value | known absent |. The same alternative rules for subsumption computation could be applied to | associated procedure | value. Similar human factor considerations are also likely to apply.

The range of procedure context values is wider and covers decision, request and intent as well as the simple observation that something was not done. Thus variants such as "not to be done" and "not requested" also need to be considered.

7.8.2.5 Terminology query languages



A terminology query language goes beyond *compositional grammar*: it supports additional criteria used to filter content that is not necessarily part of a *concept* definition.

Example use cases:

1. The department responsible for clinical research needs to create queries that will select a portion of the terminology (i.e. "all infectious diseases caused by gram negative bacteria") that are meaningful for some specific purpose.

2. A group planning a translation project needs to define queries that will extract subject-specific *refsets* to be forwarded to specialists groups.

 **Example query:** All the *subtypes* of "Clinical finding" that include a finding site *descendant* of "thorax" and the FSN matches "*pain*"

The [Query Specification Reference Set](#) can be used to store string forms of terminology queries, associated with a *reference set*; this enables the generation of its members.

An standard specification for a Query Language for *SNOMED CT* is not yet defined. Many terminology management tools have developed their own query languages and representations. The "*Refset Specifications*" in the *IHTSDO workbench* is a representative example.

Expression languages or *concept* definition representation grammars can be used as query languages, using the techniques described in [Concept definition queries](#) and [Expresion retrieval](#).

7.8.3 Creating legacy queries



A *terminology server* may also support the creation of queries that retrieve data encoded in *SNOMED International* (*SNOMED CT* version 3.x), Clinical Terms Version 3 or earlier versions of the *Read Codes*.

For example, a *terminology server* may generate a SQL predicate list that includes the *SNOMEDIDs* or *CTV3IDs* of all unique *subtype descendants* of a specified *Concept*. Some *constraints* on this functionality may be necessary as top-level or other general *Concepts* may generate extremely long lists of *descendant identifiers*.

7.9 Creating and maintaining Reference Sets



This section describes the basic steps required to create and maintain *Reference Sets*.

7.9.1 How to create a new Reference Set using an existing pattern



In order to create a new *Reference Set*, you will need access to a *namespace* in order to generate *SCTIDs*. Within your *namespace*, you should add one *moduleId* *concept* (with an *FSN* and *Preferred Term*), under the *|Module| sub - hierarchy* within the metadata, for each of your authoring organizations .

Then, follow the steps below to create a new *reference set*.

- [Define the Reference Set in the metadata hierarchy](#);
- [Define the Reference Set Attributes within the metadata hierarchy](#);
- [Create the Descriptor for the Reference Set](#);
- [Add members to the Reference Set](#).



Caution: All *components* created during these processes must have unique *Identifiers* and all those *Identifier* must be allocated in the correct partition of your organizations namespace. For details see [SNOMED CT Identifiers](#).

7.9.1.1 Define the Reference Set in the metadata hierarchy



First, create a *concept* for the *Reference Set*:

Table 263: Reference Set Management Example - Add Reference Set Concept

Field	Data type	Value
id	SCTID	A unique id in your <i>namespace</i> .
effectiveTime	Time	The nominal date of release for your <i>reference set</i> .
active	Boolean	'1'
moduleId	SCTID	The module <i>Identifier</i> for your authoring organization .

Then, add up to three *Descriptions* for the FSN, the *Preferred Term* and optionally the Purpose:

Table 264: Reference Set Management Example - Add Descriptions for Concept

Field	Data type	Value
id	SCTID	A unique id in your <i>namespace</i> .
effectiveTime	Time	The nominal date of release for your <i>reference set</i> .
active	Boolean	'1'
moduleId	SCTID	The module <i>Identifier</i> for your authoring organization .
conceptId	SCTID	The <i>Identifier</i> of the concept describing the <i>Reference Set</i> that you've just added.
languageCode	String	The <i>language</i> of the <i>Description</i> .
typeId	SCTID	Create up to three <i>descriptions</i> , with each of the following types: FSN , Synonym , Purpose . The first two are mandatory, the third is optional.
term	String	<i>Terms</i> for the FSN, the <i>Synonym</i> and the Purpose . The <i>Synonym</i> will be the <i>string</i> used to commonly refer to the <i>Reference Set</i> . The conventions for creating <i>terms</i> for the FSN and <i>Synonym terms</i> are described in Naming Conventions for Reference Sets .

Add an | is a | *Relationship* to link the *Reference Set* to the appropriate pattern:

Table 265: Reference Set Management Example - Link to Metadata for Reference Set Pattern

Field	Data type	Value
id	SCTID	A unique id in your <i>namespace</i> .
effectiveTime	Time	The nominal date of release for your <i>reference set</i> .
active	Boolean	'1'
moduleId	SCTID	The module <i>Identifier</i> for your authoring organization .
sourceId	SCTID	The <i>Identifier</i> of the <i>concept</i> describing the <i>Reference Set</i> that you've just added.
destinationId	SCTID	The <i>concept</i> describing the pattern that this <i>Reference Set</i> follows, a <i>descendant</i> of <i>Reference Set</i> in the metadata <i>hierarchy</i> .
relationshipGroup	Integer	'0'
typeId	SCTID	is a
characteristicTypeId	SCTID	Stated <i>relationship</i>

7.9.1.2 Define the Reference Set Attributes within the metadata hierarchy



Add new *concepts* for each of the *Reference Set* member attributes, if necessary. If the *Reference Set* attributes describing the pattern are adequate to describe the *Reference Set's* attributes, then these can be used instead, and you can skip to the next section.

You may wish to create your own *Reference Set* attributes for one of the following *reasons*:

- You wish to give one or more of the attributes a different name than that of the pattern;
- You wish to make the purpose of a particular attribute more explicit in the metadata;
- You wish to limit the set of allowed values for one or more of the attributes;
- You wish to make the type of one or more of the attributes more specific than that given in the pattern.

You may add new *concepts* for some of the attributes, and reuse existing *concepts* for other attributes, if you wish.

For each attribute that you wish to create, first add a *concept*.

Table 266: Reference Set Management Example - Add Reference Set Concept

Field	Data type	Value
id	SCTID	A unique id in your <i>namespace</i> .
effectiveTime	Time	The nominal date of release for your <i>reference set</i> .

Field	Data type	Value
active	Boolean	'1'
moduleId	SCTID	The module <i>Identifier</i> for your authoring organization .

Then, link it with an | is a | *Relationship* into the | Reference set attribute | metadata *hierarchy*.

Table 267: Reference Set Management Example - Link to Metadata Hierarchy

Field	Data type	Value
id	SCTID	A unique id in your <i>namespace</i> .
effectiveTime	Time	The nominal date of release for your <i>reference set</i> .
active	Boolean	'1'
moduleId	SCTID	The module <i>Identifier</i> for your authoring organization .
sourceId	SCTID	The <i>Identifier</i> of the <i>concept</i> describing the <i>Reference set</i> attribute that you've just added.
destinationId	SCTID	Reference set attribute
relationshipGroup	Integer	'0'
typeId	SCTID	is a
characteristicTypId	SCTID	Stated relationship

Then, add up to three *Descriptions* (for FSN, Preferred Term and optionally Purpose) for each of the new attributes:

Table 268: Reference Set Management Example - Add Descriptions

Field	Data type	Value
Id	SCTID	A unique id in your <i>namespace</i> .
effectiveTime	Time	The nominal date of release for your <i>reference set</i> .
active	Boolean	'1'

Field	Data type	Value
<i>moduleId</i>	<i>SCTID</i>	The module <i>Identifier</i> for your authoring organization .
<i>conceptId</i>	<i>SCTID</i>	The <i>Identifier</i> of the <i>concept</i> describing the attribute that you've just added.
<i>languageCode</i>	<i>String</i>	The <i>language</i> of the <i>Description</i> .
<i>typeId</i>	<i>SCTID</i>	Create up to three <i>Descriptions</i> for each new attribute, with the following types: Fully specified name , Synonym , Purpose . The first two are mandatory, the third is optional.
<i>term</i>	<i>String</i>	<i>Terms</i> for the Fully specified name , a Synonym and the Purpose . The Synonym will be the <i>string</i> used to commonly refer to the attribute (and therefore should appear as a column header in tables showing the <i>Reference Set</i> member records).

If any of the *Reference Set* member attributes are to be limited to a range of values, then add a *concept* for each allowed value in the range, and link the *concept* using an |Is a| *relationship* to the member attribute. Then add two *Descriptions* for the *FSN* and *Preferred Term* of each allowed *attribute value*.

In order to limit the range of an attribute, it must have a type of | Concept type component| (as held in the *attributeType* field of the Descriptor - see the next section).

For each allowed value that an attribute can take, add a *concept*.

Table 269: Reference Set Management Example - Add Allowed Attribute Value Concept

Field	Data type	Value
<i>id</i>	<i>SCTID</i>	A unique id in your <i>namespace</i> .
<i>effectiveTime</i>	<i>Time</i>	The nominal date of release for your <i>reference set</i> .
<i>active</i>	<i>Boolean</i>	'1'
<i>moduleId</i>	<i>SCTID</i>	The module <i>Identifier</i> for your authoring organization .

Then, link it with an | is a| *Relationship* into the attribute that you've just added in the | Reference set attribute | metadata *hierarchy*.

Table 270: Reference Set Management Example - Link Allowed Attribute Value to Metadata

Field	Data type	Value
id	SCTID	A unique id in your <i>namespace</i> .
effectiveTime	Time	The nominal date of release for your <i>reference set</i> .
active	Boolean	'1'
moduleId	SCTID	The module <i>Identifier</i> for your authoring organization .
sourceId	SCTID	The <i>Identifier</i> of the <i>concept</i> describing the allowed <i>attribute value</i> that you've just added.
destinationId	SCTID	The <i>Identifier</i> of the <i>concept</i> describing the attribute that you've just added.
relationshipGroup	Integer	'0'
typeId	SCTID	is a
characteristicTypeId	SCTID	Stated relationship

And finally, add two *Descriptions* for the allowed *attribute value concept*:

Table 271: Reference Set Management Example - Add Description for Allowed Attribute Value

Field	Data type	Value
Id	SCTID	A unique id in your <i>namespace</i> .
effectiveTime	Time	The nominal date of release for your <i>reference set</i> .
active	Boolean	'1'
moduleId	SCTID	The module <i>Identifier</i> for your authoring organization .
conceptId	SCTID	The <i>Identifier</i> of the <i>concept</i> describing the allowed <i>attribute value</i> that you've just added.
languageCode	String	The <i>language</i> of the <i>Description</i> .

Field	Data type	Value
<i>typeId</i>	<i>SCTID</i>	Create two <i>descriptions</i> , with each of the following types: FSN , Synonym
<i>term</i>	<i>String</i>	<i>Terms</i> for the Fully specified name and a Synonym . The Synonym will be the <i>string</i> used to commonly refer to the allowed <i>attribute value</i> (and therefore should be the one shown in pick lists used when maintaining <i>Reference Set</i> member records).

7.9.1.3 Create the Descriptor for the Reference Set



Add one record to the | *Reference Set Descriptor* | *Reference Set* describing the *referencedComponentId* attribute, and one additional row for each additional optional attribute within the *Reference Set*.

These records together describe the structure of the *Reference Set*, and are called the *Descriptor* of the *reference set*, for short. If the existing *Descriptor Template* (that describes the *Reference Set's* pattern) also adequately describes the *reference set* that you've just created, then a new *Descriptor* need not be created, and this section may be skipped.

Where a *Descriptor* is created for a new *Reference Set*, it should have the same structure (i.e. - an identical number of records, each of the same attribute type or *subtype*) as the *Reference Set Descriptor* that described the parent *Reference Set* pattern.

Table 272: Reference Set Management Example - Add Reference Set Descriptor

Field	Data type	Value
<i>id</i>	<i>UUID</i>	A unique <i>UUID</i> for this record.
<i>effectiveTime</i>	<i>Time</i>	The nominal date of release for your <i>reference set</i> .
<i>active</i>	<i>Boolean</i>	'1'
<i>moduleId</i>	<i>SCTID</i>	The module <i>Identifier</i> for your authoring organization .
<i>refsetId</i>	<i>SCTID</i>	900000000000456007 Reference set descriptor reference set (foundation metadata concept)
<i>referencedComponentId</i>	<i>SCTID</i>	Set to the <i>concept</i> describing the <i>Reference Set</i> that you've just created.
<i>attributeDescription</i>	<i>SCTID</i>	Set to the <i>concept</i> describing the attribute that you've just created, or alternatively an existing <i>concept</i> under the 900000000000456007 Reference set descriptor reference set (foundation metadata concept) metadata <i>hierarchy</i> .

Field	Data type	Value
<i>attributeType</i>	SCTID	Set to a <i>descendant</i> of 900000000000456007 Reference set descriptor reference set (foundation metadata concept) in the metadata <i>hierarchy</i> . This field describes the type of the attribute. If an attribute has been limited to a range of values, then this field must always be set to 900000000000456007 Reference set descriptor reference set (foundation metadata concept) . If a <i>Reference Set</i> is the <i>child</i> of a particular <i>subtype</i> of 900000000000456007 Reference set descriptor reference set (foundation metadata concept) , this field must be the same as or a <i>descendant</i> of the equivalent field for the more general <i>Reference Set subtype</i> .

7.9.1.4 Add members to the Reference Set



Follow the steps in the next section to maintain the members of the *Reference set*.

7.9.2 How to add, change or remove members of an existing Reference Set



You should only add members to a *Reference Set* in your organization's namespace or in the namespace of an organization that has authorized you to edit that *Reference Set* and provided you with a *moduleId* in their namespace to use for that purpose.

To add a member to an existing *Reference Set*, create a new record as follows:

Table 273: Reference Set Management Example - Member Added

Field	Data type	Value
<i>id</i>	UUID	A unique <i>UUID</i> for the new member record.
<i>effectiveTime</i>	Time	The nominal date of release that this member is to be first introduced in.
<i>active</i>	Boolean	'1'
<i>moduleId</i>	SCTID	The module <i>Identifier</i> for your authoring organization .
<i>refsetId</i>	SCTID	The id of the <i>concept</i> that describes the <i>Reference Set</i> that you're adding a member to.
<i>referencedComponentId</i>	SCTID	A reference to a <i>component</i> , of type (and possibly range) limited by the Descriptor record for this <i>Reference Set</i> with <i>attributeOrder</i> '0'.
additional field 1		An optional <i>Attribute</i> , with a value, of type (and possibly range) limited by the Descriptor record for this <i>Reference Set</i> with <i>attributeOrder</i> '1'.

Field	Data type	Value
additional field 2		An optional <i>Attribute</i> with a value, of type (and possibly range) limited by the Descriptor record for this <i>Reference Set</i> with <i>attributeOrder</i> '2'.

To delete an existing member from a *Reference Set*, create a new record as follows:

Table 274: Reference Set Management Example - Member Deleted (made inactive)

Field	Data type	Value
id	UUID	A unique <i>UUID</i> of the existing member record that you wish to delete.
effectiveTime	Time	The nominal date of release in which this member is to be deleted.
active	Boolean	'0'
moduleId	SCTID	The module <i>Identifier</i> for your authoring organization .
refsetId	SCTID	As value in existing record
referencedComponentId	SCTID	As value in existing record
additional field 1		As value in existing record
additional field 2		As value in existing record

To modify an existing member in a *Reference Set*, create a new record as follows:

Table 275: Reference Set Management Example - Member Modified

Field	Data type	Value
Id	UUID	A unique <i>UUID</i> for the existing member record that is to be updated.
effectiveTime	Time	The nominal date of release that the update is to become <i>active</i> in.
active	Boolean	'1'
moduleId	SCTID	The module <i>Identifier</i> for your authoring organization .

Field	Data type	Value
refsetId	SCTID	As value in existing record. A member cannot move from one <i>reference set</i> to another.
referencedComponentId	SCTID	As value in existing record. A member cannot change the <i>component</i> that it refers to. Instead, the existing member record should be deleted, and a new one created.
additional field 1		This field may be updated. An optional <i>Attribute</i> , with a value, of type (and possibly range) limited by the Descriptor record for this <i>Reference Set</i> with <i>attributeOrder</i> '1'.
additional field 2		This field may be updated. An optional <i>Attribute</i> with a value, of type (and possibly range) limited by the Descriptor record for this <i>Reference Set</i> with <i>attributeOrder</i> '2'.

7.9.3 How to create a new Reference Set pattern



In order to create a new *reference set* pattern, follow these steps to create a new *reference set*, with the following exceptions:

- The *concept* describing the *Reference Set* pattern should be created as an immediate *child* of the | *Reference set* | *concept*, or as a *child* of another *Reference Set* pattern.
- The *Descriptions* of *typeid* | *Synonym* | and |*FSN*| should be of the form:
 - *My pattern nametype*;
 - *My pattern name type reference set* (*foundation metadata concept*).
- A Descriptor Template must be created for a pattern, following the steps as described to create a Descriptor for a *Reference Set*.

7.10 Terminology Server Software



This section outlines the possible characteristics of software that provides *Terminology services* through a programmable interface. Such software represents an approach to development that may enable more rapid implementation of *SNOMED CT*.

This guide does not specify a particular *Application Programming Interface (API)* for accessing *SNOMED CT* services. Instead it sets out the general principles and options for delivery and use of a *terminology server*.

7.10.1 Terminology server functionality



A *terminology server* should be able to deliver all the essential *Terminology services* identified in the *Terminology Services Guide (RF2) (7)*. It should also provide the recommended *Terminology services* and should achieve a performance that meets the more general requirements for the functionality of *SNOMED CT enabled applications*.

Terminology server may provide two types of service:

- Reference Services (see [Figure 122](#)):
 - Services that do not include a *user interface*;
 - The client application may use reference services to undertake many different functions;
 - For some of these functions the client application will populate an appropriate *user interface component*.

 **Example:**

A reference server may return a list of *Descriptions* matching a particular search *string*. The client application may use this data to populate a list from which a user makes a selection.

- *User Interface (UI) Services* (see [Figure 123](#)):
 - Services that include the one or more *user interface components* that can be used in and programmatically accessed by the client application.

 **Example:**

A *UI* server may provide a control that includes a text box and a list. When the user types in the text box, the server populates the list and allows the user to select an item. The selected item is accessible from the client program.

- One possible type of *UI* service is a *SNOMED CT browser* with an *API* for returning selected data to a client application:
 - This may be useful as mechanism for providing some *SNOMED CT* capabilities to an application. However, it is less suitable for frequent entry of *SNOMED CT* encoded information.

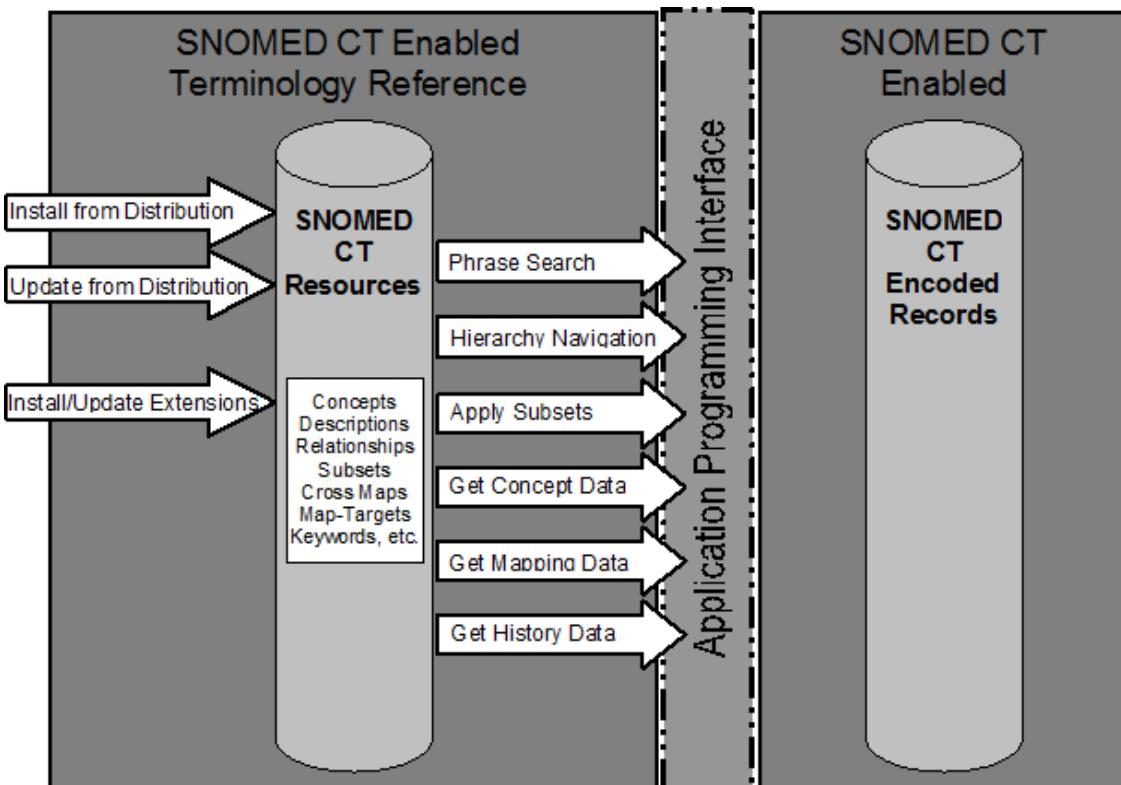


Figure 122: Terminology server providing reference services to a client application

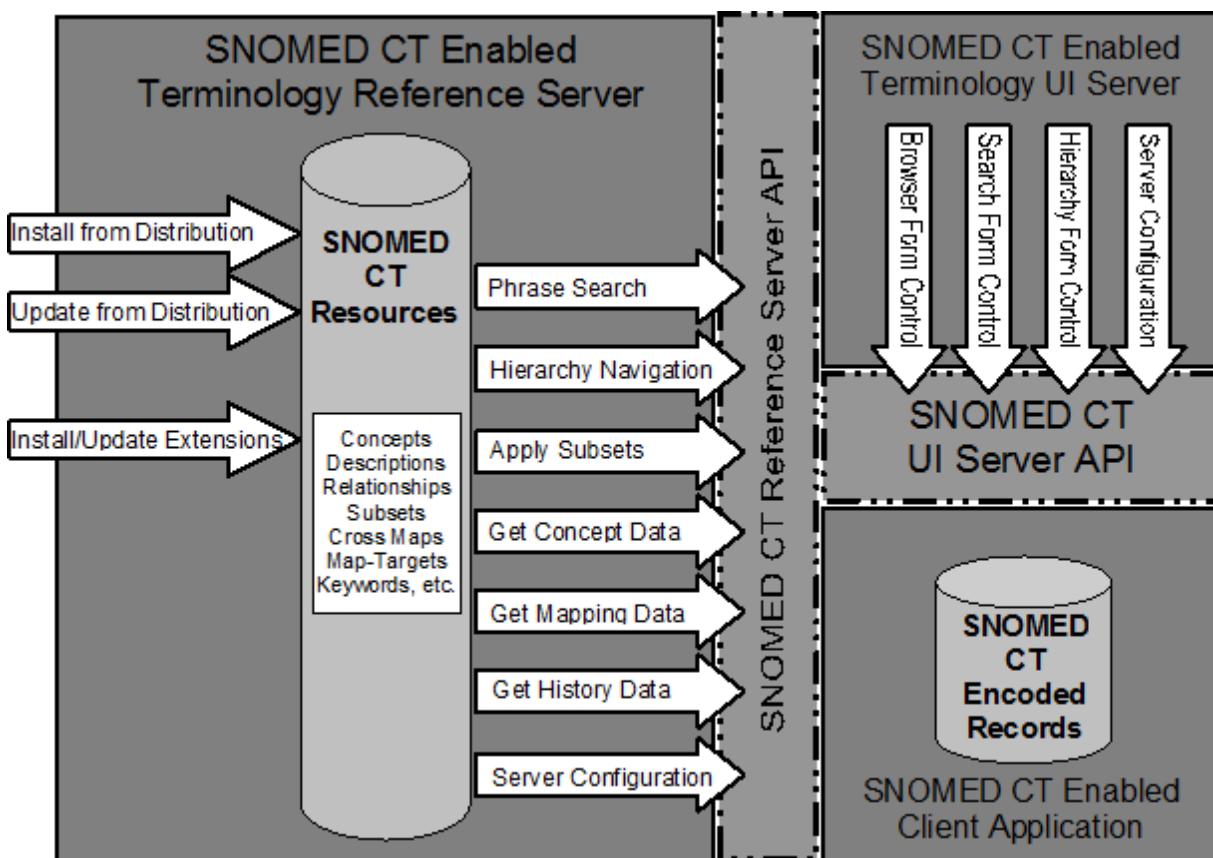


Figure 123: Terminology server providing user-interface and reference services to a client application

7.10.2 Terminology server APIs



This guide does not specify a particular *API*. The services specified in this guide may be delivered using various types of interfaces based on a range of different technologies including:

- Web services such as WSDL (Web Services *Description Language*) or REST (Representational State Transfer) interfaces;
- Java components such as JavaBeans™ or Eclipse plug-ins;
- Microsoft .NET® or Active -X® / COM / DCOM in Microsoft Windows® environments;
- CORBA® (Common Object Request Broker Architecture).

Decisions on which technologies to support depend on the intended functionality, performance, accessibility, ease of use and support requirements for maintenance or updates.

Over the past two decades there have been various efforts to specify standards for *terminology servers* and related *APIs*. The most recent development in this area is centered around the Common *Terminology Server Release 2 (CTS2)*. The requirements initially identified and documented within *HL7* have now led to an *OMG* (Object Management Group) proposal. At least one of the responses to this proposal focuses directly on *SNOMED CT* related requirements. The *OMG* process is expected to result in a detailed specification and prototype implementation during 2011.

Chapter

8

8 Record Services Guide



The following sections discuss requirements for *record services* that support entry, storage, retrieval and communication of *SNOMED CT* encoded information. The services are illustrated by [Figure 124](#).

The primary use of *SNOMED CT* is to enable information to be entered in a health record and stored in a manner that enables selective retrieval. Effective selective retrieval is required to support aggregation, analysis and decision support. Information in a health record may need to be communicated in the interests of the patient or to enable larger scale aggregation and analysis. Communication of information should convey the information expressed using *SNOMED CT* expressions in ways that preserve the semantics and thus enable recipient systems to process the information effectively.

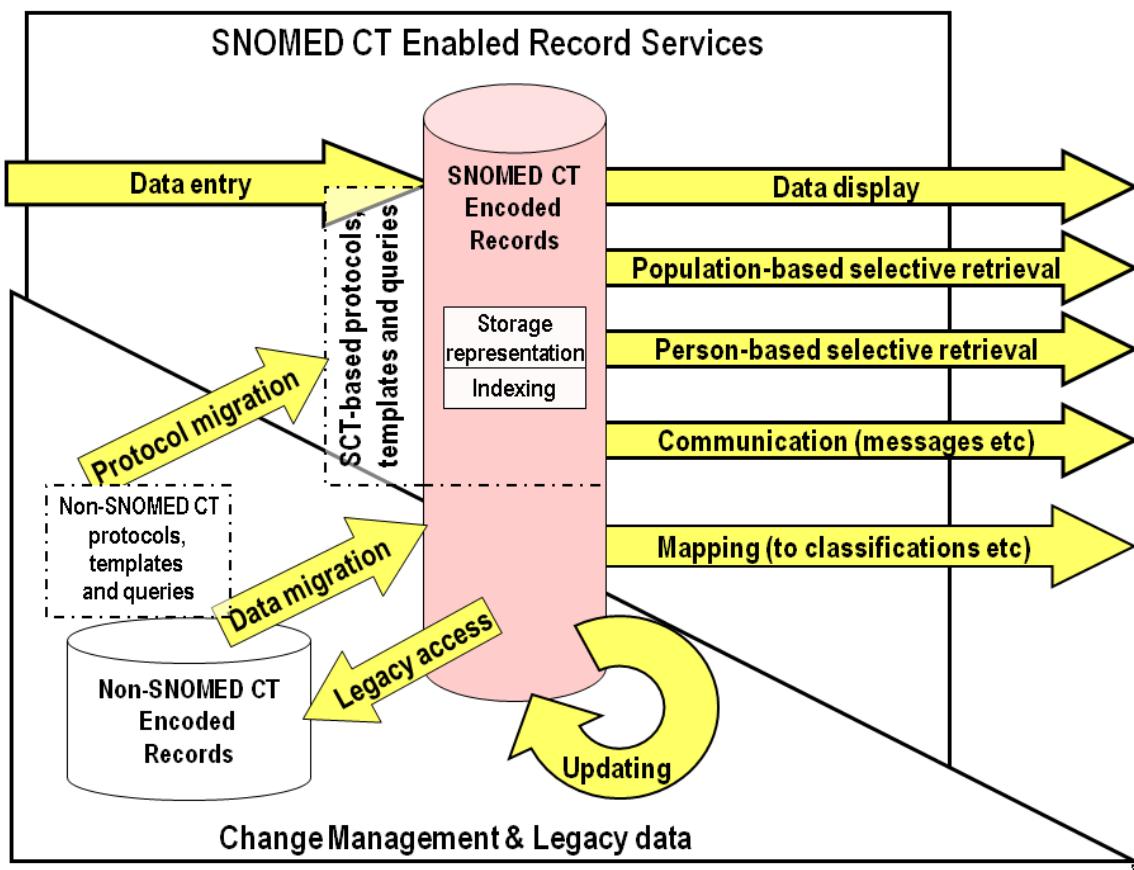


Figure 124: SNOMED CT Enabled Record Services

8.1 Entering Expressions



SNOMED CT enabled applications must facilitate the entry of *SNOMED CT expressions* in ways that allow users to capture relevant information easily and accurately. This section considers various methods that may be used to enter *SNOMED CT* information into a record. These data entry methods require the *Terminology services* specified in the [Terminology Services Guide \(7\)](#).

8.1.1 Using text searches and subtype hierarchy navigation



8.1.1.1 Selection in a browser



The starting point for a consideration of data entry is an efficient method for performing text searches and *subtype hierarchy navigation*. When these functions are integrated in a terminology *browser*, it is possible to select a *Concept* by text search and then to refine or generalize the selection to identify a more appropriate *Concept* for recording.

A terminology *browser* built into an application or offering a programmable interface can be used to allow a user to select a *Concept* (and/or *Description*) and enter this into a record.

This method of data entry allows unconstrained selection of any *Concept* from *SNOMED CT*. This can sometimes be useful but such an unrestricted method should only be used as a fallback, when more selective approaches cannot be used.

8.1.1.2 Limitations of simple browsers



A general-purpose *browser* capable of searching and navigating through the *SNOMED CT hierarchy* is a simple starting point. However, this approach is unlikely to meet the requirements for anything other than occasional entry of *SNOMED CT* encoded information. More selective mechanisms tailored to particular data entry contexts are likely to be more usable and may promote more consistent data recording.

In most situations in which clinical data is entered, access to the full content of *SNOMED CT* through a simple search and *hierarchy browser* is unlikely to be necessary and may be cumbersome and unhelpful. The main reason for this relates to the size and structure of the terminology. As a result:

- Many *terms* may match a single word or short phrase resulting in a long list of options;
- The depth and breadth of the *subtype hierarchy* and *navigation* may require selection of choices from several screens to locate the required *Concept*.

There are many ways to improve and simplify *SNOMED CT* data entry. Some of these can be used in a wide range of situations. Others are specific to constrained contexts that occur in structured data entry driven by a template or protocol.

8.1.2 Optimizing searches for data entry



8.1.2.1 Extending searches and limiting duplication



The *Terminology services* guide addresses ways of:

- Extending text searches to include similar words and phrases by making use of the *Word Equivalents Table*;
- Rationalizing text searches, which, in a simple search, return the same *Concept* more than once due to multiple matching *Terms*.

These techniques may be used to improve access to *Concepts* during data entry.

8.1.2.2 Searches with qualifier resolution



When typing text for a search, the user is unlikely to know if their intended entry can be represented by a single *Concept* or requires a *postcoordinated expression* involving additional *Concepts* or *qualifiers*. Where searches fail to find a *precoordinated* match, expansion of the search to support appropriate or commonly used *qualifiers* is likely to enhance usability.

Some *terminology servers* may provide a general facility of this type. Alternatively, a limited facility for recognizing commonly qualifying words may be used. For example, words such as "left," "right," "routine," | emergency | and | severe | are applicable as *qualifiers* when not included in a *precoordinated Concept*.

8.1.2.3 Real time searching



Conventional text searches require the user to decide how many words to enter and then explicitly request a search. When a search fails to find any matches or returns a very long list of matches, the user is obliged to repeat the process. The need to undertake this type of user interaction for every coded entry is likely to create a significant disincentive to effective data entry.

One possible solution to this is an interface that performs real - time checking of the number of matches as the user types. The interface may indicate this to the user, allowing them to decide when to stop typing and commence the search. A further enhancement is to automatically return the list of matches whenever the user stops typing, or when the number of matches reduces to an acceptable level.

8.1.2.4 Background encoding



Techniques that support real-time searches and *qualifier* resolution may also be extended to enable background encoding of complete sentences as they are entered. This method can be applied to text entered by typing or by voice recognition.

As text is entered, the search mechanism attempts to narrow the selection. If this process eventually finds a single good match, this is used to encode the text. The match should be displayed allowing the user to override it, but the default action is to accept the encoding. If at the end of a sentence there are multiple possible matches, then these are presented for user selection.

There are many possible variants on this technique. For example, as the possible matches are narrowed down, the system could offer an auto-completion option similar to that used in web browsers and word-processors.

Caution: Anyone implementing this approach should take care to undertake appropriate quality assurance of the results. Mention of this approach to data entry does not imply that it is considered safe for a given use-case. Formal professional assessment of the risks and benefits of any type of automated encoding is essential.

8.1.2.5 Automatic and semi-automatic encoding



Techniques similar to those used for background encoding can be applied to previously entered text or to text entered by voice recognition or optical character recognition. Where such methods are used there is likely to be a need for manual intervention to resolve uncertain encoding. The requirement for manual intervention will depend on the sophistication of the matching techniques and the extent to which accuracy is safety-critical. If encoded data is to be used by clinical decision support protocols, which may influence the treatment of a patient, extreme care is needed when using automatic encoding and tools that allow manual review are essential. A less rigorous approach may be acceptable where the purpose of encoding is for aggregation and analysis of large volumes of population data.

Caution: Anyone implementing this approach should take care to undertake appropriate quality assurance of the results. Mention of this approach to data entry does not imply that it is considered safe for a given use-case. Formal professional assessment of the risks and benefits of any type of automated encoding is essential.

8.1.2.6 Mnemonics and personal favorites



Groups of people, such as practitioners of a discipline or specialty, frequently use similar sets of *Descriptions* and *Concepts*. Lists of widely understood (or easily learned) abbreviations or mnemonics that allow rapid entry of these commonly used *concepts* are recommended as a way of accelerating repetitive recording.

A similar facility may also be useful for individual users or organizations that have sets of *Descriptions* and *Concepts* that they use frequently. An easy way to use options to store and recall personal favorites with user-defined abbreviated access *terms* will enhance usability and significantly increase the speed of data entry.

User guidance may be necessary to minimize the risk of shortcuts such as these being overused. Unless the general search facilities are also easy to use, it is likely that users will favor the shortcuts even when it would be more appropriate to use a more accurate but less accessible *Concept*. An unchecked bias toward easy to record *Concepts* may lead to deterioration in data quality, statistical anomalies, and in the worst case, inappropriate treatment.

8.1.3 Constraining searches for data entry



8.1.3.1 Constraining searches by status



Searches should usually be filtered according to the *status* of the *Concept* and/or *Description*.

An application should only allow *Active Concepts* and *Descriptions* to be entered in a patient record. *Active Concepts* and *Descriptions* include those with the *status* "current" or "pending move."

There are a few cases where a user may legitimately wish to search *Inactive Concepts* and *Descriptions*. Possible reasons for this include creating or editing queries that locate previously entered data recorded using *Concepts* and *Descriptions* that are no longer recommended for *active* use. Therefore, it must be possible to disable or vary the *status* filters applied to searches.

8.1.3.2 Constraining searches by subtype ancestors



Searches may usefully be limited to *Concepts* that have a specified *supertype ancestor*, which is appropriate for the context of a particular field, template or protocol.

Example:

When attempting to record the diagnosis "renal calculus," it is not helpful for a search to include the procedures that may be carried out to treat a renal calculus.

8.1.3.3 Constraining searches by Reference Sets



Searches for *Descriptions* or *Concepts* may need to be constrained by *Reference Sets*.

Applications should allow searches to be filtered, ordered or otherwise prioritized in accord with one or more *active Reference Sets*. Specifically, the search mechanism should support the following functions with respect to the following types of *Reference Sets*:

- Filtering of search and *navigation* results to include only those *Descriptions* that are referenced by the Language *Reference Sets* may be applied to limit a search to those *Descriptions* applicable in a particular *language* or *dialect*.
- A Simple *Reference Set* may be used to filter, sort or highlight the results of text search or hierarchical *navigation*. This may simplify or encourage selection of *Concepts* or *Descriptions* used in a particular country, organization or specialty.
- A Simple *Reference Set* or an Ordered *Reference Set* may be used to specify or *order* the valid *Concepts* for entry in a particular field.

8.1.4 Constraining and extending hierarchical navigation for data entry



8.1.4.1 Using the subtype hierarchy for data entry



The most visible hierarchical construct in *SNOMED CT* is the *subtype hierarchy*. This is constructed using a set of logical rules. The purpose of this *hierarchy* is to support data retrieval and aggregation by addressing the question "is concept -A a subtype of concept -B."

The same *hierarchy* can be used for data entry *navigation* but it is not designed for this purpose. Its depth and breadth are determined by logical rules of subsumption rather than by usability. As a result:

- There is no upper limit on the number of *subtypes* a *Concept* may have. This is true because there is no rule that determines the number of *subtypes* that a real world *concept* may have. However, long lists of options are not conducive to effective data entry.
- There is no fixed limit to the number of hierarchical steps between a generalized *Concept* and its most refined *subtype*. This is true since there is no preordained limit on the extent of possible *refinement* of a real world *concept*. However, data entry procedures that involve stepping through several levels of choices before reaching the required selection impair usability.
- The *subtypes* of a *Concept* do not have any particular *order*. The | is a | *Relationship* is primarily a property of the *subtype Concept* and does not express an ordinal position. This is true because logical *subtypes* are inherently an unordered set. However, a user is likely to find it easier to locate their required selection if members of hierarchical lists are displayed in some recognizable *order*.
- The issues of depth, length and *order* noted above are also subject to change between releases. The addition of an intermediate *Concept* or reclassification after the addition of new *defining characteristics* will introduce new layers in the *hierarchy*. Some *Concepts* will then move from the list of immediate *subtypes* of a *Concept* to become *subtypes* of a more refined *Concept*. Hierarchical changes may sometimes simplify *navigation* by reducing the number of choices at a given hierarchical level. However, the general effect of improvements in the *subtype hierarchy* will be to increase its depth and thus to increase the number of steps from a particular general *Concept* to its most refined *subtypes*.
- The nature of a *subtype hierarchy* means that there may be many routes from a given *Concept* to its more general *descendants*. This means that some of the choices presented for user selection are redundant since they simply offer alternative routes to the same *Concept*.

Routine use of *subtype hierarchy navigation* is not recommended for data entry. However, despite the drawbacks listed above, the *subtype hierarchy* may be useful for undertaking an exhaustive search for a particular refined *Concept*.

Example:

The *Concept*"Laparoscopic emergency appendectomy" can be reliably located by *subtype navigation* from any of its supertypes: "appendectomy," "laparoscopic appendectomy" or "emergency appendectomy."

8.1.4.2 Using Ordered Reference Sets to support data entry



Ordered Reference Sets provide alternative hierarchical representations of *SNOMED CT*. They are intended to support data entry by addressing the limitations of the *subtype hierarchy* discussed in the previous section.

- Usability *constraints* can be placed on the number of levels in the *hierarchy* and the number of options displayed at each level in the navigational *hierarchy*:
 - If there are relatively few options and many layers, the most common options can be brought to a higher level.
 - If there are long lists of options, these may be subdivided with less frequent options moved to lower levels.

- Options that are rarely or never used by a particular user community can be excluded from a navigational *hierarchy* to limit the range of choices. According to requirements, these options may remain accessible by switching to a *subtype* view.
- Options at each hierarchical level can be ordered to meet the expectations of users and/or to facilitate rapid access to commonly used options.
- The available options at a particular level can be kept stable across releases without affecting the accuracy of the *subtype hierarchy*.

An Ordered *Reference Set* may be based on the foundation of the *SNOMED CT subtype hierarchy*. This can then be modified to add ordering and other features discussed above. An alternative starting point is a *hierarchy* of classification derived from another coding scheme or classification.

 **Note:**

An Ordered *Reference Set* derived from the *Clinical Terms Version 3 hierarchy* is provided with the *SNOMED CT Developer Toolkit* as an example.

Alternative Ordered *Reference Set* can be created from scratch (or as variants of a common source *hierarchy*) to provide views to support users with different requirements. Since *Navigational Hierarchies* do not affect interpretation, retrieval and aggregation, data entered in using different views can be analyzed consistently.

8.1.5 Constraining data entry



Some *Concepts* or *Descriptions* displayed by searches, hierarchical *navigation* or other methods of data entry may not be suitable for recording in a patient record. Various *reasons* for this are discussed in the following sections. They include:

- Status of the *Concept*
- Status of the *Description*
- Special *Concept*
- *Subtype* relevance;
- *Reference Set* inclusion;
- Cross Mappability;
- Context.

8.1.5.1 Constraining data entry by Concept Status



 **Note:** In *Release Format 2* the active field is used to determine whether a *concept* is intended for active use. The additional status information described below is available if required in the *Concept Inactivation Reference Set* and Historical Association from inactive to active *concepts* are found in a *Historical Association Reference Set*.

Inactive Concepts should not be added to a record. *Inactive Concepts* include those with the following status values:

- **Retired:** The *Concept* should not be used. No further information is available.
- **Duplicate:** The *Concept* was found to be the same as another *Concept*. The duplicated *Concept* can be identified by following | SAME AS |Relationship and this may be used.
- **Outdated:** The *Concept* that is no longer meaningful due to changes in accepted understanding of biology, disease, health or related subject areas.
- **Ambiguous:** The meaning of the *Concept* is ambiguous because its associated *Descriptions* have different meanings. The *Concepts* representing possible alternative meanings can be identified by following the | MAY BE A |Relationships and one of these may be selected for use.
- **Erroneous:** There is an error in the representation of the *Concept*. The corrected *Concept* can be identified by following the "REPLACE BY" |Relationship and this may then be used.

- **Moved Elsewhere:** The Concept is now maintained in a different namespace. The target namespace can be identified by following the | MOVED TO |Relationship. If release data for this namespace is available the relevant Concept can be identified by locating a | MOVED FROM |Relationship in the target namespace that refers to the original Concept. The Concept from the target namespace may then be used. If the release data for the target namespace is not available, the Concept should not be used.
- **Limited:** Classification Concepts with limited semantic stability (e.g. terms that contain "not otherwise specified," "NOS," "not elsewhere classified," "NEC").

Active Concepts include those with different *ConceptStatus* values and depending on circumstances, some of these may not be appropriate to add to a record:

- **Current:** Suitable for general use.
- **Pending move:** Suitable for use unless and until the Concept is available in another namespace. Concepts with this status will generally become *inactive* with the status "Moved elsewhere" (see above) in a subsequent release.

8.1.5.2 Constraining data entry by Description Status



👉 **Note:** In Release Format 2 the active field is used to determine whether a *Description* is intended for active use. The additional status information described below is available if required in the *Description Inactivation Reference Set* and references from inactive to active description are found in a *Historical Association Reference Set*.

It is not appropriate to allow *Inactive Descriptions* to be added to a record. *Inactive Descriptions* include those with the following *DescriptionStatus* values:

- **Retired:** These should not be entered in a record. No further information is available.
- **Duplicate:** The *Description* contains the same *Term* and the same *Concept Identifier* as another *Description*¹⁹. The duplicated *Description* should be identified either:
 - By following the appropriate Reference;
 - By string matching among the other *Descriptions* of the associated *Concept*.
- **Outdated:** A *Description* that contains an obsolete *Term* which refers to a valid *Concept*. Another *Description* associated with the *Concept* should be used.
- **Erroneous** - There is an error in the representation of this *Description*. The corrected *Description* can be identified by following the appropriate Reference and this may then be recorded.
- **Inappropriate:** The *Term* in this *Description* should not be associated with this *Concept*. The same *Term* associated with its appropriated *Concept* can be identified either:
 - By following the appropriate Reference.
 - By searching for the same *Terms* associated with as *Active Description*. More than one *Active Description* may be associated with a *Term* if that *Term* is used to describe more than one *Concept*.
- **Concept retired:** The *Description* is associated with an *Inactive Concept*. The *Description* was *active* until the *Concept* was inactivated. The *Description* should not be used and is retained only to provide valid *Descriptions* associated with the *Concept* and any legacy data previously recorded using this *Concept*.
- **Moved elsewhere:** The *Description* is now maintained in a different namespace. This applies where a *Concept* has also been moved to the new namespace (see above).

8.1.5.3 Excluding Special Concepts and Model Components



Concepts that are subtypes of the top-level Concept 370115009 | Special concept | (RF1) or 900000000000441003 | SNOMED CT Model Component | (RF2) are rarely if ever required in clinical end-user searches. Therefore, they should be excluded from text searches except where explicitly needed to meet a

¹⁹ Note that Duplicate Terms associated with different Concepts may be Active Descriptions and these are not marked with the *DescriptionStatus* "duplicate." See also Duplicate Terms Subsets.

particular requirement (e.g. to display a | Namespace concept |, a | Linkage concept | or a | Navigational concept |).

8.1.5.4 Constraining data entry according to subtype relevance



It may be necessary to prevent entry of a *Concept* in a *subtype hierarchy* that is inappropriate to a particular data entry field or to a particular part of a patient record. For example:

- An application should not allow a disorder *Concept* to be recorded in a field intended for recording a procedure (or vice-versa);
- An application should not allow a *Concept* that is a *subtype descendant* of the top-level *Concept "attribute"* to be recorded, except to associate another *Concept* with an appropriate qualifying value;
- An application should not allow a *Concept* that is a *subtype descendant* of the top-level *Concept "qualifier value"* to be recorded, except where it qualifies an appropriate attribute *Concept*.

8.1.5.5 Constraining data entry using Reference Sets



In some cases, identifying selected portions of the *SNOMED CT hierarchy* may be a sufficient *constraint* for entering data into a record. However, that is not always sufficient if *Concepts* from multiple hierarchies are required, or if there is a need to hone down the entry options from the full *hierarchy*. To meet these requirements applications should allow data entry to be constrained by *Reference Sets*.

Applications should be able to:

- Permit or prevent the entry of *Concepts* or *Descriptions* that are members of a specified Simple *Reference Set*.

Example:

A UK GP system might:

- Prevent the entry of *Concepts* in a Simple *Reference Set* that contains all *Concepts* that are non-human;
 - Enable the entry of *Concepts* in the "UK Administrative *Reference Set*" only when entering information in an administrative context.
-
- Encourage or inhibit the entry of *Concepts* or *Descriptions* according to their order in an Ordered *Reference Set*.

Example:

A specialty system might prompt for confirmation when the user records a procedure not in a specified specialty Simple *Reference Set*.

8.1.5.6 Constraining data entry based on cross-mappability



One of the requirements for some applications may be that the data recorded in particular fields has to be mapped to a particular classification or grouping scheme.

One way to simplify this process is for the application to check mappability at the time of data entry. If a selected *Concept* has no unambiguous map, the application may encourage or compel the user to refine their selection until a mappable *Concept* has been selected.

This type of facility should not be applied in situations where it may inappropriately affect the perceived accuracy or detail of a clinical record.

8.1.5.7 Constraining data entry based on context



All fields (data elements) used for data entry must be analyzed to understand what underlying context is implied. The appropriate *concepts* should then be selected for the *value set* of each field. *Concepts* from the Clinical Findings, Procedures, and Observable entities hierarchies can be used directly if the default assumptions are true. Otherwise, *concepts* from the *Concept -Dependent hierarchy* should be selected.

Particular care must be taken with systems that enable *postcoordinated* constructs to ensure that the appropriate context attributes are included.

To precoordinate concepts that do not already exist in *SNOMED CT*, care must also be taken to determine if any axis modification is shifting the meaning of a *concept* so it should move to the *Situation with explicit context hierarchy*.

8.1.5.8 Absolute and configurable constraints



Some of the *constraints* on data entry discussed in the preceding sections are absolute while others should be configurable.

- An application should not allow *Inactive Concepts* or any *Special Concepts* to be recorded.

Constraints based on *subtype hierarchies*, *Subsets* or *Reference Sets* and *Cross Maps* should usually be configurable to particular institutions, users and/or data entry fields.

8.1.6 Configuring and applying data entry constraints



The previous sections describe various mechanisms for extending and constraining search and *navigation* during data entry. The scope of applicability of these facilities varies and these variations affect the way in which they may be implemented.

A few *constraints* apply to all data entry events in a particular application. These fixed *constraints* could be hard-coded in the application or explicitly optimized when importing and indexing *SNOMED CT* content.

👉 Example:

One example is to exclude *Inactive Concepts* and *Descriptions* from searches. Before building this type of facility into an application, care should be taken to consider circumstances, such as creation and editing of queries where access to *Inactive Concepts* may be required.

Most search *constraints* are to some extent configurable and these require greater flexibility in the application design. There are several types of configurability that may be required. These range from installation configuration to context-specific dynamic configuration.

8.1.6.1 Installation configuration of data entry



Requirements of an organization that are general to all users may be applied when installing the application or when importing or indexing *SNOMED CT* content. These may include:

- Language *Reference Set* which constrain searches to the local *language* and *dialect*.
- Simple *Reference Sets* which apply national or organization *constraints* applicable to all users of the application.
- Simple *Reference Sets* which apply *constraints* applicable to all the clinical disciplines or specialties that use the installed system. For example, installations that are not intended for use in veterinary medicine will apply *Reference Sets* that exclude specific veterinary *Concepts* and *Descriptions*.
- An Ordered *Reference Set* that provides a data entry *hierarchy* appropriate to the needs of all users within an organization .

8.1.6.2 Log-on configuration of data entry



The application should allow search *constraints* that are specific to a particular user or group of users when loading or logging on to an application. The range of possible search *constraints* may be preset at installation but it should be possible to apply the user profile *constraints* without a significant delay. Uses of this type of configuration include:

- Simple *Reference Sets* which apply *constraints* or optimizations applicable to a particular specialty;
- Ordered *Reference Set* that provide a restricted or extended data entry *hierarchy* appropriate to the needs or preferences of a particular specialty or user;

- Language *Reference Set* that meet the needs of particular users in a multi-lingual environment.

Consideration should be given to requirements for this type of search configuration to be modified by a user or system administrator.

8.1.6.3 Dynamic reconfiguration of data entry



Constraints that assist fast and consistent routine data entry may sometimes need to be relaxed to enable more complex entries to be made.

- If a *Ordered Reference Set* limits the scope of hierarchical *navigation*, the application should enable the user to utilize the *subtype hierarchy* to allow other options or a more complete set of options to be reviewed;
- If a user is unable to locate the *Concept* that they require, it may be useful to enable some or all of the search *constraints* to be temporarily lifted.

8.1.6.4 Context-sensitive of data entry constraints



Some *constraints* may apply to particular data entry contexts. To support this type of functionality, an application should be able to switch between sets of search *constraints* in real - time. The *constraints* need to change instantly as a user moves between different data entry fields. Context-dependent *constraints* may include:

- Limitation of a search to *subtype ancestors* of an appropriate *Concept*.

Example:

A field for entry of a procedure may be associated with a *constraint* that limits searches to *subtypes* of the *Concept* "procedure."

- Limitation of a search to the *Concepts* or *Descriptions* that are members of an appropriate *Simple Reference Set*.

Example:

A field for entry of a laboratory service request may constrain searches to a list of valid investigations supported by a particular laboratory.

- Use of a particular *Ordered Reference Set* or an specified sub-branch of an *Ordered Reference Set*.
 - This is an alternative approach that may be used to allow more sophisticated control of data entry in a particular context.

8.1.7 Entering qualifiers and other postcoordinated representations



SNOMED CT contains many *precoordinated Concepts* that allow fairly complex *Concepts* to be represented by a single *Concept Identifier*. It also permits the qualification or *refinement* of *Concepts* to represent more detailed *Concepts* by *postcoordinated* combinations of several *Concept Identifiers*.

Several types of *postcoordinated* data are outlined in this section from the perspective of data entry. These include *refinement*, qualification and combination. The requirements for and relevance of each of these will depend on decisions about data representation within patient records.

8.1.7.1 Entering refined defining characteristics



The application may allow a user to refine a *Concept* by selecting a *subtype* of one of its *defining characteristics*.

 **Example:**

One of the *defining characteristics* the *Concept* | total replacement of hip | is | using | = | hip prosthesis. | The *Concept* | total replacement of hip | could be refined by allowing the user to specify one of the *subtypes* of "hip prosthesis."

Refinement options may be entered by selecting from hierarchical lists showing *subtype* values for each of the refinable characteristics. Simple lists or option buttons could support selection from limited sets of possible *refinements*. Wider ranges of potential *refinement* could be facilitated by text searches constrained to *subtypes* of one or more of the refinable characteristics.

Refinement should not be allowed for *defining characteristics* with the *refinability* value "not refinable."

 **Caution:**

Some concepts should not be refined if the result means the new concept is not a subtype of the parent concept.

This situation occurs when context such as "Family history" or | Planned Procedure | is attached to a Clinical Finding or Procedure. "Family history" of a Clinical Finding needs to be defined in the *situation with explicit context hierarchy*. All *postcoordinated* constructs should consider the impact of context.

8.1.7.2 Entering sanctioned qualifiers



The application should allow a user to qualify a *Concept* by selecting one of its *qualifying characteristics*. These are represented by *Relationships* with a *CharacteristicType* value that indicates that they are optional qualifications rather than *defining characteristics*.

Entry of *qualifiers* may be supported by allowing selection from simple lists or by appropriate sets of option buttons. Automatic selection of *qualifiers* by parsing a search *string* may reduce the need for direct entry of some *qualifiers*.

The application should support *refinement* of *qualifying characteristics* in accordance with the *refinability* field value. The *refinability* value "mandatory to refine" indicates that entering the unrefined *qualifier* adds no useful information. In these cases, the application should force *refinement* of the selected *qualifying characteristic* to one of the *subtypes* of the *qualifier* value.

8.1.7.3 Entry of unsanctioned qualifiers



The application may also permit *refinement* of a *Concept* by the addition of *qualifiers* that are not sanctioned by inclusion as *qualifying characteristics* in the *Relationships Table*. These *qualifiers* may be constructed using pairs of *Concepts*, a *subtype* of the top-level *Concept* "attribute" combined with an appropriate value.

Any facility to allow qualification of *Concepts* in this way carries a risk of creating nonsensical or contradictory statements. It may also result in incomplete or inappropriate retrieval where the *qualifier* significantly affects the meaning of the *Concept*.

8.1.7.4 Constraints on the entry of qualifiers



Qualifiers should only be used where the result of applying them results in a true *subtype* of the original *Concept*. Therefore, *qualifiers* should **not** be used for the following purposes:

- Negation.

 **Example:**

| Fracture of humerus | should not be qualified by "excluded."

It would be inappropriate for data retrieval to treat this as a *subtype* of the clinical finding | Fracture of humerus |.

- Certainty.

 **Example:**

| Carcinoma of cervix | should not be qualified by "possible."

It would be inappropriate for data retrieval to treat this as a *subtype* of the diagnosis of | Carcinoma of cervix |.

- Subject of information.

 **Example:**

| Diabetes mellitus | should not be qualified by "family history."

It would be inappropriate for data retrieval to treat this as a *subtype* of the diagnosis of | Diabetes mellitus | in the patient.

- Planning stage.

 **Example:**

"Hip replacement" should not be qualified by "planned" or "requested."

A count of "Hip replacement" operations performed should not include this. Decision support protocols should not assume the patient has had this operation.

These and similar major modifications need to be handled in ways that are explicit and ensure that queries and decision support protocols are able to accurately retrieve and analyze the available information.

8.1.7.5 Entry of concepts combinations



The application may allow other combinations of *Concepts* in a single statement where a *Concept* that represents the full scope of an activity is not available. This approach might for example be applied where a single procedure, which lacks a *precoordinated SNOMED CT* representation, is a combination of two procedures that can be separately represented in *SNOMED CT*.

Facilities for entering combined *Concepts* should be implemented and used with care. It is appropriate to use these facilities when the combined result is conceived as a single statement that could potentially be used in many different patient records.

 **Example:**

A diagnosis of "gallstones with cholecystitis" could be entered by selecting the "gallstone (disorder)"|235919008and then selecting | cholecystitis (disorder) |76581006and combining these in a single statement²⁰.

It is not appropriate to use these constructs to attempt to express an entire encounter, episode or clinical history in a single statement.

 **Example:**

If a patient is treated for "gallstones with cholecystitis" diagnosed by "ultrasonography of biliary tract" with a course of | amoxycillin | followed after the acute phase has resolved by a | cholecystectomy |, this should **not** be entered as a single complex *postcoordinated* statement combining the diagnosis, investigation and treatments.

8.1.8 Template and protocol driven data entry



In many healthcare disciplines similar data sets are collected for each patient. Clinical consultations for many conditions involve repeatable sequences of data entry. These structured and predictable data entry

²⁰ There is also a precoordinated Concept "Calculus of gallbladder with cholecystitis" which is equivalent to this postcoordinated combination.

requirements can be met using sets of customized data entry fields or forms (templates) designed to collect particular data items. These data entry templates may be presented in a predefined sequence, as selected by the user. Alternatively the sequence of data entry may follow a branching pathway with previously entered data determining which branches are taken (protocols).

When using a structured data entry mechanism, *SNOMED CT* encoded data can be selected in a variety of ways. Some of these involve direct selection of *Concepts* and *Descriptions* while in others the encoding may result from responses to simple choices or entry of particular data values. The following list outlines some of the possible mechanisms for *SNOMED CT* encoding during structured data entry:

- User selection from a small list of possible *Descriptions* applicable to a particular field in a template or step in a protocol:
 - A Simple Reference Set with *Descriptions* as members or a Language Reference Set may specify the set of applicable *Descriptions*.
- Text search limited to a set of *Concepts* applicable to a particular field in a template or step in a protocol:
 - A Simple Reference Set may specify the set of applicable *Concepts*;
 - Alternatively the applicable *Concepts* may be specified as the *subtype descendants* of a single *Concept*.
- Association of a *Concept* with particular options presented by a check box, option button or other data entry control:
 - When selections are made using this control the appropriate *Concept Identifier* is added to the record.
- Association of a *Concept* with a data control used for entering a numeric or other value:
 - When a value is entered in this control it is labeled with the appropriated *Concept Identifier*.
- Association of a *Concept* with a particular combination of values or the result of a computation involving several items of previously entered data:
 - In its simplest form this is an *extension* of one or both of the previous options;
 - In some applications, information derived from the user-entered data, by decision support tools, may be encoded in this way.

Some installations allow free text to be entered at point of care if a needed *concept* is not included in the predefined short list. This text is then reviewed by trained staff who can then search and find the appropriate *Concept*, request the addition of a new *Concept*, and/or request that the *Concept* be added to the template's short list for future use. The success of this option relies upon trained staff who are available to do the review on a timely schedule, and the willingness of the clinician to use this approach sparingly, as it is greatly preferred to choose the appropriate *concept* and not enter free text.

8.2 Storing Expressions



SNOMED CT enabled applications must support the storage of *SNOMED CT expressions* in ways that represent relevant information within the record system. This section is concerned with the different approaches that may be used to store *expressions* in ways that enable them to be subsequently retrieved, displayed, processed and communicated.

The term *SNOMED CT expressions* includes also single *Concept Identifier expressions*, that identify only one specific *concept*. Even when *postcoordination* is not supported in an implementation, each time a single *Concept Identifier* is being assigned to a clinical record it represents a way of using a *SNOMED CT expression*.

8.2.1 precoordinated and postcoordinated representations



8.2.1.1 precoordination



The simplest form in which any *concept* can be stored is as a single *Identifier*. This is referred to as a *precoordinated expression*, because all aspects of a potentially multifaceted *concept* are *precoordinated* into a single discreet form.

SNOMED CT contains more than a quarter of a million *concepts*, and thus allows a wide range of clinical statements to be expressed in *precoordinated* form.

👉 **Example: Laparoscopic emergency appendectomy - precoordinated**

A *precoordinated expression* 174041007 | laparoscopic emergency appendectomy | can be used to record an instance of this procedure.

The procedure "Laparoscopic emergency appendectomy" has at least three distinct facets: "removal of appendix", "using a laparoscope" and "as an emergency procedure". *SNOMED CT* includes a *concept* that precoordinates these facets.

The *concept* 174041007 | laparoscopic emergency appendectomy | has the following *defining characteristics*:

260870009 | priority | = 25876001 | emergency |,

116680003 | is a |=80146002 | appendectomy |,²¹

425391005 | using access device | = 86174004 | laparoscope |.

8.2.1.2 postcoordination



A multi-faceted *concept* can be stored using a combination of *Identifiers* for its individual facets. This is referred to as *postcoordination*, because the various aspects of the *concept* are coordinated during data entry rather than in the preparation of the terminology. Three types of *postcoordination* are described in the following sections.

8.2.1.3 postcoordination by refinement



Refinement is a type of *postcoordination* in which a *concept* is made more specific by refining the value of one or more of the defining attributes of the *concept*.

👉 **Example: Total replacement of hip using a Sheehan total hip prosthesis - postcoordinated**

A *postcoordinated expression* based on the *concept* 52734007 | total hip replacement | can be used to record an instance of this procedure. The definition of this *concept* includes 363699004 | direct device |=304120007 | total hip replacement prosthesis | and the value of this attribute can be refined to 314580008 | Sheehan total hip prosthesis | (which is a *subtype* of 304120007 | total hip replacement prosthesis |). Therefore, the following *postcoordinated expression* can be created and used to represent this procedure:

52734007 | total hip replacement | : 363699004 | direct device |=314580008 | Sheehan total hip prosthesis |.

Another common use of refinement is to represent a situation such as a family history, or a planned procedure. In this case, a *concept* representing the general type of situation can be refined by applying a clinical finding or procedure.

²¹ In practice the relationship 116680003 | is a |=80146002 | appendectomy | is represented via intermediate supertype and is also represented by the following defining characteristics 260686004 | method |, 405813007 | procedure site - Direct | = 66754008 | appendix structure |.

 **Example: Family history of temporal arteritis - postcoordinated**

A *postcoordinated expression* based on the *concept* 281666001 | family history of disorder | can be used to record a family history of any disorder. The definition of this *concept* includes 246090004 | associated finding |=64572001 | disease | and the value of this attribute can be refined to 400130008 | temporal arteritis | (which is a *subtype* of 64572001 | disease |). Therefore, the following *postcoordinated expression* can be created and used to represent this family history:

281666001 | family history of disorder | :246090004 | associated finding |=400130008 | temporal arteritis |.

8.2.1.4 postcoordination by qualification



Qualification is a type of *postcoordination* in which a *concept* is made more specific by applying value to attributes that are permitted by the *Concept Model*. Unlike refinement, the attributes applied need not be present in the definition of the *concept* that is being qualified.

 **Example: Laparoscopic emergency appendectomy - postcoordinated**

A *postcoordinated expression* based on the *concept* 80146002 | appendectomy | can be used to record an instance of this procedure by separately specifying the access instrument and priority. The *concept* 80146002 | appendectomy | does not have defined values for the attributes 260870009 | priority | and 425391005 | using access device | but the *Concept Model* permits these to be added to *subtypes* of 71388002 | procedure |. Therefore, the following *postcoordinated expression* can be created:

80146002 | appendectomy |:260870009 | priority |= 25876001 | emergency |, 425391005 | using access device |= 86174004 | laparoscope |

This *postcoordinated expression* is equivalent to the definition of the *concept* 174041007 | laparoscopic emergency appendectomy |. However, the *postcoordinated* approach can also be applied to procedures for which there is no *precoordinated concept*.

8.2.1.5 postcoordination by combination



 **Example:**

"Gallstones with cholecystitis" could be represented by combining the *concepts* for the disorders "gallstones" and | cholecystitis | as a single *postcoordinated statement*. Neither of these *concepts* is really a *qualifier* of the other since it could equally well be regarded as | Calculus of gallbladder with cholecystitis |.

SNOMED CT allows *Concepts* to be combined in *postcoordinated statement*.

Combinations like this should only be used to represent *concepts* that can be regarded as discreet reusable clinical statements. They should not be used to construct arbitrarily complex representations of multiple statements to a particular record.

Some *concepts*, such as the first and last examples above, can be represented in either a *postcoordinated* or *precoordinated* form. However, there are other *concepts*, like the second example above, for which no *precoordinated Concept* exists in *SNOMED CT*. Although future releases of *SNOMED CT* will include new *precoordinated Concepts*, there will always be some clinical *Concepts* that require *postcoordination*.

8.2.1.6 Representing postcoordination



This guide does not specify a single right way to represent *postcoordinated expressions*.

Alternative representations have different profiles of advantages and disadvantages. The choice of representation depends on functional requirements including performance, information model of the software application and the communication standards to be supported.

Some alternative representations are summarized below. These summaries illustrate some of the main options and do not go into extensive technical detail. Detailed design may lead to further alternatives that are not documented here.

Each of the following summaries assumes that *SNOMED CT expressions* are stored in (or associated with) one or more fields within particular types of record entry. The *expression* is only one part of the data in that record entry.

8.2.1.6.1 Parsable text representation



A way to represent *postcoordinated SNOMED CT* information as a simple parsable text *string* is summarized below:

- Each clinical statement is recorded as a row in a relational database table (or as an element in an XML document);
- The schema for representation of clinical statements contains a field (or element) for representation of the *SNOMED CT expression*;
- The *expression* field (or element) contains a text *string* that is formatted in accordance with the *SNOMED CT compositional grammar*.

8.2.1.6.2 Unrestricted relational representation



An unrestricted relational database representation of a *postcoordinated expression* requires that a data item that may be expressed using *SNOMED CT* is modeled in a way that permits an indeterminate number of *attribute-value pairs* to be appended to a *focus concept*. In addition, the value within each *attribute-value pair* must be able to be refined by addition of nested *attribute-value pairs*.

This offers a flexible and extensible approach but adds significantly to database design complexity. Disadvantages arising from this complexity include storage capacity requirements and the impact on writing queries and retrieval performance.

8.2.1.6.3 Restricted relational representation



An alternative restricted relational representation of *postcoordinated SNOMED CT* information is summarized below:

- Each clinical statement is recorded as a row in a relational table.
- The clinical statements table contains a field for a *Concept Identifier*.
- The clinical statements table also contains fields for a specified number of *qualifiers*. These fields may be provided in different ways:
 - Each *qualifier* is represented by two *Concept Identifier* fields (one for the attribute and one for the value) and an optional field for *Relationship group* field. With this option the only restriction is the total number of *qualifiers* or modifiers that can be stored for each *Concept*.
 - Each *qualifier* is represented as a single *Concept Identifier* and carries the value of a *qualifier* attribute specific to that field. This restricts the usable *qualifiers* to those specified in the database schema.
 - Similar to above, but with different sets of qualifying attributes available according to the semantic type of the primary *Concept* in the statement. There are various ways of implementing this approach to ensure that the appropriate interpretation is applied to each row of the table.
- Combined *Concepts* may be represented by explicitly combining two rows of the clinical statements table.

Unlike the representations discussed in previous subsections, this approach limits the expressivity of *postcoordinated statements*. The advantage of this restricted approach is that it reduces the number of joins involved in retrieval queries. In some software environments this may significantly improve performance.

The balance between demands for flexibility and performance depends on user requirements. Therefore, limitations in expressivity may be acceptable for some users or user communities but not for others. However, it should be noted that these limitations might cause difficulties when communications are received from systems that support richer forms of *expression*.

8.2.1.6.4 XML Representations



A way to represent *postcoordinated SNOMED CT* information as an XML element is summarized below:

- Each clinical statement is recorded as a row in a relational table or as an element in an XML representation.
- The clinical statements table (or element) contains a field (or element) for representation of the *concept*.
- The *concept* field (or element) contains an XML *expression* that encapsulates a *postcoordinated* representation of the *concept* according to a parsable syntax specified for this purpose:
 - Various alternative XML representations could fulfill this role.

8.2.1.6.5 Representation as precoordinated content



In some implementations, *expressions* are stored as *precoordinated content*, with new *concepts*, *Descriptions* and *Relationships* in an extension namespace.

User input includes also a text label for the *expression*, and the new *concept* is created, usually a team of expert *SNOMED CT modelers* review the new *concept* for quality assurance. Other implementations requires that user enter only the text label, and then the modelers team can associate the label to an existing *concept*, or create a new *concept* in a local extension using the label as a *Description* and adding the new *Relationships* for the *concept* definition.

This approach is called Managed Content Additions (MCA). Has some advantages like having all new content available for text searches by users, and allowing the use of a *description logics* classifier for inferring *Relationships* and super-types, avoiding the need of complex real-time *expressions* computations. On the other having a centralized team of experts represents an expensive approach and a possible bottleneck for terminology development, as the experts need to review all content additions in the system.

8.2.1.7 Storing and retaining original expressions



Transforming an *expression* to a *normal form* may be necessary to support effective data retrieval.

However, even quite small minor corrections to the definition of a *concept* in future releases may significantly alter the resulting *normal form* of the same *expression*.

Therefore, it is recommended that:

- The primary or original record should be stored using the representation that is as close as possible to the form in which it was recorded.
- If *transformations* to alternative representations are used to enhance the efficiency of retrieval, these should be stored as secondary supporting tables or indices:
 - This has the advantage that these alternative forms can be regenerated based on the most up to date set of definitions when a new release of *SNOMED CT* is installed, without affecting the integrity of the original records.

8.2.2 SNOMED CT storage issues for electronic health records



8.2.2.1 Storing Concepts in electronic health records



Information in an *electronic health record* should accurately reflect the way it was recorded by its author. If the author of a statement in the clinical record chooses a particular form of representation the system should faithfully store the information in that form.

Following this principle, the recommended approach for representation of *SNOMED CT* in a *electronic health records* is as follows:

- If, during data entry, an author selects a single *precoordinated Concept* to represent a clinical statement, the *Identifier* of that *Concept* should be stored in the record:

- This form of representation should remain as the original record of that statement. It should not be replaced by an apparently equivalent *postcoordinated transformation* of this *Concept*.
- If, during data entry, an author constructs a clinical statement by selecting a *Concept* and one or more *qualifier values*, *refinements* or additional *Concepts*, the *Identifier* of all the relevant *Concepts* should be stored in the record in a manner that reflects the *relationships* between them:
- This form of representation should remain as the original record of that statement. It should not be replaced by an apparently equivalent *transformation* of this *Concept* into a *precoordinated* or differently constructed *postcoordinated* form.

An application should prompt for author endorsement of any alternative form of representation that it proposes to store in the original *electronic health record*. In this case, if the author accepts the alternative form presented by the application, this form should be stored as the original record.

The forms in which a technical implementer may wish to store data for efficient retrieval may differ from the forms dictated by the principles appropriate to storage of original entries in a *electronic health record*. However, it is recommended that any retrieval- oriented representation should be derived from rather than replace the original form of the record.

8.2.2.2 Storing terms



A *electronic health record* should also store the *terms* that were actually displayed to and selected by the author of the record. In some *Realms* the *Description Identifier* may be regarded as an adequate proxy for the full representation of the associated *Term*. However, in other jurisdictions there may be a requirement to store the original text as entered or selected by the user.

Storing the *Description Identifier* has the added advantage if a *Description* is found to be wrongly associated with a particular *Concept* or if the associated *Concept* is found to have non-synonymous *Descriptions*. In these cases, the *Description Identifier* can be used to map the information to the appropriate disambiguated *Concept*.

8.2.2.3 Maintaining integrity following SNOMED CT releases



A *SNOMED CT release* may contain changes to that state of one or more *Concepts* or *Descriptions* referenced by a stored *expression*. The original recorded form of each stored *expression* should be retained as record of the information actually entered. However, it may also be useful to include updated representations that take account of changes to the referenced *SNOMED CT* content.

Release Format 2 files contain previous states of each component allowing comparisons to be performed. In addition, members of an appropriate | Historical association reference set | allow data originally recorded with a *Concept* that has been marked as *Inactive* to be mapped to an appropriate *Active Concepts*.

If clinical records are updated using this history information, the changes should be appended to the original representation, rather than replacing it. This ensures that any changes arising from a subsequent release can apply the improved mapping to the original *Concept* this can be utilized to enhance data quality.

 **Note:** In *Release Format 1*, *SNOMED CT Component History*, *Reference* and *Relationship* tables contain information that allows data originally recorded using *Inactive Concepts* to be appropriately mapped to *Active Concepts*.

8.2.3 SNOMED CT storage options for effective retrieval



The form in which records are represented may have a substantial impact on the efficiency, accuracy and completeness of retrieval. The forms that best suit retrieval may differ from the forms that are required to meet the principles of clinically safe and legally valid *electronic health records*.

8.2.3.1 Storing information as entered



This option leaves information in the form entered in the *electronic health record* with no additions to assist future retrieval. The application must do all the work needed to locate the required records and compute subsumption and *equivalence* when a request is made to retrieve data.

8.2.3.2 Using an Expression Repository



An innovative approach to the issues raised by literal storage of *postcoordinated expressions* is to implement an *expression repository*. Each unique *expression* used in the system is stored in a referenced database table and assigned an internal unique *Identifier* (e.g. a *UUID*). When an *expression* is used in a clinical record entry the unique *expression id* is used to reference the *expression* in the repository.

The key advantages of this approach of this approach are:

- The *expression Identifier* can have a fixed size whereas a *postcoordinated expression* is of variable and indeterminate size. This significantly improves storage and index efficiency.
- The *expression repository* can also be used to store *normal form* representations of each *expression* and to relate these to the original *expression*. This optimizes performance for *expression normalization* during retrieval.
- The *expression repository* could also be processed by a *Description Logic Classifier* and a *transitive closure* table of all the *expression* used in the application could then be generated. *postcoordinated* retrieval would then be highly optimized by using the *transitive closure* to test a single join between each predicate and the candidate *expressions*.

8.2.3.3 Maximizing postcoordination



One possible approach to optimization of retrieval is to *transform* the original stored information into an equivalent representation with the minimum number of *postcoordinated* components.

The objective of this approach is to allow the generation of simple indices for the *precoordinated* representation. It is then possible to undertake most retrievals using the *|* is a *| subtype hierarchy* to compute whether *Concepts* in the record are *subtypes* of the *Concepts* used to specify retrieval. Where *postcoordination* is required, the minimum number of additional tests are required to confirm that a *Concept* in the record meets the specified retrieval criteria.

One difficulty with this approach is that there may be more than one representation that requires the same degree of *postcoordination*. This is discussed in more detail and illustrated in [Transforming expressions to normal forms](#).

If this approach is adopted additional rules need to be applied to determine the choice between alternatives with a similar number of *postcoordinated* components.

Example:

In the hypothetical example illustrated in [Figure 30](#), the *Concept* "red steel pedal bicycle", for which no *precoordinated* representation exists, could be represented as:

"red pedal bicycle" + | make of | = | steel |

or

"steel pedal bicycle" + "color" = "red"

Both are equally close to the objective of minimizing *postcoordination*. A rule is needed to determine which of these is preferred. There is no obvious right or wrong solution to this but a simple rule that places the attributes in an *order* will, if applied consistently, allow all *postcoordinated* representations to be reduced to a single minimized form.



8.2.3.4 Maximizing postcoordination

An alternative approach is to expand any *precoordinated concepts* in the record to their fullest possible *postcoordinated forms*. This general type of *transformation* is illustrated in [Transforming expressions to normal forms](#).

This approach requires a richer record structure but has the advantage that there are three possible end-points to *postcoordination*, each of which ensures that any computably equivalent representations of *Concepts* will expand to an identical *postcoordinated form*. The three end-points are summarized here:

- Short canonical form:
 - This is the most parsimonious of the three options.
 - A *concept* is represented as the combination of:
 - *Subtype relationships* with its most proximate *primitive supertypes*;
 - The recorded *qualifier values* and/or *defining characteristics* that distinguish it from its most proximate *primitive supertypes*.
- Long canonical form:
 - This option is more verbose as it includes some redundancy.
 - A *concept* is represented as the combination of:
 - *Subtype relationships* with its most proximate *primitive supertypes*;
 - All of its recorded *qualifier values* and/or *defining characteristics*, irrespective of whether they are shared by its most proximate *primitive supertypes*.
- Exhaustive postcoordinated form:
 - This option is extremely verbose.
 - A *Concept* is represented as a combination of:
 - *Subtype relationships* with all of its *supertype ancestors*
 - All of its recorded *qualifier values* and/or *defining characteristics*, irrespective of whether they are shared by its most proximate *primitive supertypes*.

If the retrieval criteria are expressed in a similar form, a relatively simple *query* can interrogate the record for all entries with a matching set of *primitive Concepts* and specified characteristics.

8.2.4 Record architectures, structures and semantics



8.2.4.1 Record structure standards and proposals



SNOMED CT is a controlled terminology that can be used in many different health record systems. The semantic model of *SNOMED CT* does not replace the need for a logically sound health record structure. Furthermore, the *IHTSDO* does not specify a particular health record structure for use in conjunction with *SNOMED CT*. However *SNOMED CT* representations of clinical *concepts* are intended to meet the needs of standard health record architectures for a consistent controlled coded terminology.

In particular, there is a strong interest in co-evolution of *SNOMED CT* and the following standards to provide a strong standard semantic foundation for future *electronic health record* development.

- The healthcare communication and structured document standards of *HL7* (www.hl7.org). In particular:
 - The *HL7 Reference Information Model* and the associated development methodology;
 - *Release 2* of the *Clinical Document Architecture (CDA)*;
 - The *Guide to the Use of SNOMED CT with HL7 Version 3* developed by the *TermInfo Project*.

- The European (CEM) Standard for Electronic Healthcare Record Communication (*EN13606*) (www.centc251.org).
- Continuing development and adoption of these Standards at the International level within ISO TC215.

8.2.4.2 Using SNOMED CT in standard architectures



The broad principles of the established health record architectures are based on a layered structure of components that contain and provide context to lower level components.

The container structures include some or all of the following:

- A top-level component representing the entire health record of one person.
- Intermediate layers representing information from various sources.
- A fixed transaction/composition layer at which an entry or set of entries are attributed to (and possibly signed by) an author:
 - Examples of this level include consultation notes, letters, reports, and other documents.
- Further levels that represent logical grouping within a record covering:
 - Topics, heading and categories;
 - Cluster or batteries of closely associated information.

Within the containment structures are two lower level components:

- Clinical statements:
 - A clinical statement may vary in structure to accommodate different kinds of information (e.g. patient history, clinical finding, investigation results, plans, procedures, medication and other therapies).
- Link statements:
 - Link statements state associations between clinical statements.
 - Links statements can be used to specify:
 - Problem-oriented groups of record components and viewing;
 - Causal and other specified links recorded by the author of a record entry.

Each health record component has the potential to include:

- Dates and times of actual and planned events.
- Associations with people, organizations , devices and other entities that participate or are used in relations to a recorded event or plan.
- Codes or other representations that name or provide the semantic information container, link, or statement:
 - *SNOMED CT* fulfills this role in a structured health record.
- Additional data including text, numeric values, images and other digital data.

When *SNOMED CT* is used in a structured record, the links and temporal associations of components combined add further richness to the potential power of *expression*. This has significant advantages and is essential for many types of aggregation and decision support. However, it also adds a complicating factor that should be taken into account when designing, recording, storage, and retrieval facilities.

Example:

To retrieve and analyze the records of patients with two potentially related conditions such as "AIDS" and "Gastro-enteritis" it is not necessary for this combination to be represented in a single *precoordinated* or *postcoordinated concept*. Instead, it is possible to look for co-existence of the individual *Concept* "Gastro-enteritis" within the records of patients who also have "AIDS."

- The advantage of this is that there is no need for the clinician to have made the association between the two conditions. Therefore a more complete assessment of the incidence of "Gastro-enteritis" in patients with "AIDS" can be made.
- The disadvantage is that if a *precoordinated* or *postcoordinated* SNOMED CT representation of the combined *concept* is used, these records will not necessarily be computably equivalent to those with the two conditions recorded separately.

There is no absolute rule on when to use multiple statements associated using record structure constructs, and when to use intrinsic *precoordinated* or *postcoordinated* SNOMED CT representations. The decision maybe influences the functionality of a particular system and the specific user requirements that the system is serving. However, the following guideline is suggested:

- A combined *precoordinated* or *postcoordinated* representation is appropriate if:
 - The combined *concept* is a discrete recognizable *Concept* that differs in some way from the simple combination of the two *concepts*. For example:
 - | Diabetic cataract | is not the same as | Diabetes mellitus | +| Cataract | because other types of cataract may occur in the same patient;
 - | Fracture of radius and ulna | is a clinically recognizable injury, which is most effectively conveyed as a single *concept*.
- Separate records for each *Concept* are appropriate if any of the following apply:
 - The combined *Concept* represents the coincidence of two potentially associated conditions or procedures;
 - The temporal and other characteristics of the two *Concepts* are different;
 - Where the association between the two *Concepts* is causal.

 **Example:**

| Fracture of femur | caused by "fall down stairs" should be represented as separated statements linked by an appropriate record structure component. The SNOMED CT Concept "Due to" could be used to name the link between these statements.

8.2.5 Safely representing the context of recorded codes



A variety of contextual factors may affect the interpretation of statements. Contextual factors typically fall into the gray area between record structure and the semantic model. Some of these may have a profound impact on the meaning or interpretation of a statement.

This section divides this issue into four distinct categories:

- Contextual information that is not represented by SNOMED CT
- Structures that may be labeled using SNOMED CT
- *Status terms* that have a profound effect on SNOMED CT encoded statements;
- Context that can safely be represented using *qualifiers*

8.2.5.1 Contextual information that is not represented by SNOMED CT



Clinical statements that contain SNOMED CT *Concept* representations will be associated with some information which is not intended to be represented using SNOMED CT:

- Dates, times of an activity of recording and activity;
- Quantitative information including ranges and durations;
- *Identifiers* or names of authors, providers of information or other parties involved in a recorded activity.

SNOMED CT is not intended to represent this information. Appropriate constructs in a standardized or proprietary record architecture should be used to relate this information to *SNOMED CT* encoded clinical statements.

8.2.5.2 Structures that may be labeled using *SNOMED CT*



Clinical statements may be contained within structures that represent collections of related information. According to the nature of these structures, *SNOMED CT* may be used to label them. However, care should be taken to ensure that any semantic implication from such a label is clearly specified.

Many labels (such as headings within a document) may be used only to organize information for a human reader. The existence of a label such as "plan" or "family history" (even if encoded using *SNOMED CT*) may not necessarily affect the computer interpretation of the data within it.

Implementers should take extreme care to ensure that any semantic implication that a human reader may assume from such labels is stored on the system in a manner that allows safe interpretation. It is recommended that any apparent inherited semantic context should be represented explicitly at the individual statement level.

Example:

If a data element "Family history" is used and the *concept* | diabetes mellitus | is encoded under that heading, the statement stored in the record should encapsulate the full semantics (i.e. Family history+(Associated finding=Diabetes mellitus) using the *SNOMED CT Concept Model*.

Other areas in which structures might be labeled with *SNOMED CT Concepts* include:

- Links between statements - *SNOMED CT* could be used to identify the nature of the link.

Example:

To indicate a presumed causal *relationship*.

- Indication of types (rather than identities) of people or organizations .

Example:

To indicate that the source of a piece of information was the patient themselves or a specified relative.

8.2.5.3 Context and Axis Modifiers



The following are examples of "axis modifiers" which may fundamentally alter the interpretation of information encoded using *SNOMED CT* :

- Subject of information.

Example:

Stating that a relative of the patient has a particular disease. This may be recorded to state either "family history" or a | social context |. If the disease is represented as a *SNOMED CT Concept* then it must be distinguishable from a statement that patient has that disease.

- Stage.

Example:

Stating that a patient should have, has been referred for, or has declined to undergo a particular procedure. These must be distinguishable from statements that a patient has had the stated procedure.

- Negation and uncertainty.

Example:

Stating that a diagnosis has been excluded or is unlikely.

- Contra-indication.

 **Example:**

Related to a treatment specified using its *Concept Identifier*.

There is a temptation to use these modifications as though they were *qualifiers*. This is not a safe practice because the assumption is that a *qualifier* refines the meaning of the qualified *Concept*. A refined *Concept* should always be a *subtype* of the original *concept*. This is not the case for these major modifications as illustrated by the following:

- Records of a | family history of | +| Diabetes mellitus | would not be expected in a response to a *query* for patients with a record of diabetes mellitus (and its *subtypes*);
- Records of "planned" + "hip replacement" should not be counted when analyzing the records of patients who have had any type of "joint replacement.";
- Records that state "meningitis" + "excluded" should not be counted as cases of meningitis;
- Records that state the patient is | allergic to | + "penicillin" are not records of treatment with an antibiotic.

This issue is discussed in more detail in the section on data entry.

The recommended approaches are:

- To ensure that the record structure captures this information in a consistent and reliable way that can be interpreted accurately when retrieving or communicating information.
- If *postcoordinated SNOMED CT* representations are used, the situation *concept* should be qualified by the finding or procedure:
 - For example, the following condition is valid | family history of disorder |: | associated finding | = | diabetes mellitus |
 - The finding or procedure must **not** be qualified by the situation as this would result in an *expression* that computed as a *subtype* of the clinical *concept*.
 - For example, the following *expression* is **not** valid | diabetes mellitus |: | qualified by | = | family history |.

8.2.5.4 Context that can safely be represented using qualifiers



Where a contextual modification can be logically regarded as a *refinement* of the original *Concept* it is reasonable to use a *qualifier*. Examples of this include "severity," "episodicity" and "laterality."

8.2.5.5 Concepts with built-in context



Some *Concepts* derived from earlier terminologies (i.e. *SNOMED International* and the *Read Codes*) contain built-in context. An example is the *concept* "FH: Diabetes mellitus" (FH being an abbreviation for family history). These *concepts* are in the *Situation with explicit context hierarchy*.

To the extent possible with released context attributes, these *Concepts* are defined (and will continue to be reviewed) so that they are computably equivalent to appropriate *postcoordinated* representations.

 **Example:**

The *concept* "FH: Diabetes mellitus" is defined to be equivalent to the *postcoordinated* representation²²

- "family history" + (| associated finding | = | diabetes mellitus |).

²² This is not the case in the first release as the appropriate defining characteristics are not in the release set.

8.3 Retrieval and Aggregation



SNOMED CT allows consistent processable representation of clinical information. *SNOMED CT enabled applications* should harness this capability to with practical tools for selective retrieval of information in individual clinical records. They should also support aggregation and analysis of clinical data derived from populations of records.

8.3.1 Requirements for selective retrieval



Selective retrieval is an essential function for a health record system. There are two main types of requirements:

- Retrieval of selected records from the records of members of a population of patients for one or more purposes, including the following:
 - Aggregations and analysis of data to support:
 - Epidemiological monitoring;
 - Clinical research;
 - Audit of care delivery;
 - Service planning.
 - Identification of patients with specific risk factors or other characteristics:
 - To allow specific preventative, investigative or therapeutic measures to be appropriately focused;
 - To allow further selective retrieval and analysis of the records of a subpopulation of patients;
 - To enable selection of patients for entry in a clinical trial.
- Retrieval of selected records from an individual patient record to enable:
 - Display of summary views and/or pre-completed template screens containing appropriate selected information.

Example: Active problems/diagnoses, current medication, recent investigation results or blood pressure readings.
 - Automating responses to questions posed by a decision support protocol.

Example: To check the record for specified symptoms, findings, investigations, procedures or diagnoses.

The following subsections address issues and requirements that are common to all types of retrieval.

8.3.1.1 Retrieval performance



The following sections identify factors that may influence performance when undertaking selective data retrieval. There are no fixed rules for optimization of retrieval performance. Application developers should interpret the issues outlined in the guide in the light of their experience with the operating systems and data management tools that they use.

An evaluation of different approaches to retrieval was undertaken for the *NHS*, in connection with work on *Clinical Terms Version 3* implementation. This showed that the "best" approach was not the same for all relational databases. Some software environments favor one approach while a different approach may be more effective in another environment. Therefore, it is likely that some of the factors discussed will have a significant impact on some developers, while being less relevant in others.



8.3.1.2 Retrieval quality

The quality of selective retrieval is measured in terms of two factors:

- **Completeness:** Retrieval should select all records that meet the selection criteria;
- **Specificity:** Retrieval should not select any records that do not meet the selection criteria.

The semantic structures of *SNOMED CT* assist application developers to achieve these goals by allowing different *expressions* that represent the same or similar information to be recognized and compared (see [Supporting Selective Data Retrieval](#)).

The meaning of a *SNOMED CT expression* may be modified by the context in which it is used. Aspects of this context are represented by:

- The record structure in which the *expression* is stored.
- Data directly associated with the *expression* in the record structure (e.g. dates and times, numeric values and units, the identity and role of the originator of the information or the performer of a procedure).
- Explicit or temporal associations with other information in the same record (e.g. co-existent conditions, likely causes of an abnormal observation, reasons for an investigation or therapeutic intervention, etc).

Storing similar information in differing structures complicates retrieval since each query must take account of alternative ways in which the required information may be stored. As a result the semantic strength of *SNOMED CT* may be obscured by the varied approach to structure. Therefore, realization of the full potential benefit of *SNOMED CT*, requires an information model that accommodates *SNOMED CT expressions* and ensure consistent storage of similar information.

Another limiting factor for retrieval is the consistency and completeness of recording. The extent of use of *SNOMED CT* depends in part of policy and guidance at national or organizational levels, which in turn depends on requirements and priorities for data retrieval and reuse. From a technical implementation perspective a key factor in delivering consistent retrieval is a user-interface that facilitates, simplifies and encourages consistent data entry which uses *SNOMED CT expressions* to the extent need to meet relevant requirements.



8.3.1.3 Retrieval criteria involving record structure

Before addressing the specifics of *SNOMED CT* related retrieval criteria it is important to recognize that these only form one part of the picture. Most selective retrieval criteria will include a mixture of predicates, some of which apply to *SNOMED CT* encoded data and some of which apply to other data in the patient record. This non - *SNOMED CT* encoded data includes:

- Data directly related to coded clinical statements. This includes:
 - Dates and times (e.g. time of an event of finding).
 - Organizations , people or devices involved in a recorded activity or finding.
 - Temporal or causal *relationships* with other clinical activities or findings.
 - Quantitative values associated with *SNOMED CT* encoded statements.
 - Associated *status* and contextual information.
- Data related to the patient:
 - Age and sex;
 - Organizations and people responsible for care;
 - Occupation, pre-existing disorders or other known risk factors.

The interplay of these factors with *SNOMED CT* encoded data may affect the optimum approach for data retrieval. Some non - *SNOMED CT* encoded retrieval criteria may significantly reduce the potential set of patients or in patient record entries that qualify for retrieval. In these cases, it may be useful to apply these criteria before testing *SNOMED CT* specific criteria.

 **Example:**

- A retrieval request for the rubella vaccination *status* of eight-year-old girls in a family practice with average population distribution requires the review of less than 1% of the population of records.
- A retrieval request for patients who have undergone a particular procedure in the last month only needs to review record entries made in the last month.
- A retrieval request for the most recent investigation results and *current* medication might be more processed by initially identifying a set of recent records. Checking these records for relevant *SNOMED CT* values may be more efficient than applying individual queries to the entire record for each of the required items of recent information.
- A retrieval request for people with a rare clinical condition, who also have a relatively common disorder, may be more efficient if the few people with the rare condition are selected first, limiting the scope of the *query* for the more common condition.

These examples illustrate a general point rather than to offer guidance on the specific searches. It is important to bear in mind that the performance, completeness and specificity of retrieval are dependent on the structure of the record as well as the semantics of *SNOMED CT*.

8.3.2 Retrieving records containing selected concepts and their subtypes



Information in health records may be expressed at various levels of specificity.

 **Example:**

To represent diagnoses of:

- Chest infection;
- Left lower lobe pneumonia caused by pneumococcus.

Criteria for selective retrieval may also need to be stated to different levels of detail.

 **Example:**

To retrieve all records of

- Respiratory tract infections;
- Left lower lobe pneumonia;
- Pneumococcal pneumonia.

Occasionally a *query* may be designed to retrieve only record entries that include a particular general *Concept*. This may be useful for a quality review or to find record entries that are too general to *cross map* to a required classification.

However, in most cases, a general *query* should include more specific *Concepts* recorded in the record. For example, if the selected *Concept* is | Respiratory tract infection | the user would expect record entries containing *Concepts* such as | Chest infection | or "Left lower lobe pneumonia caused by pneumococcus" to be retrieved. The *subtype hierarchy* of *SNOMED CT* is designed to facilitate this type of retrieval. Four techniques that can be used for this purpose are outlined in the following subsections.

 **Note:**

The *subtype hierarchy* is improved with new releases of *SNOMED CT*. These changes need to be considered if more than one version of the hierarchies is used for data analysis.

8.3.2.1 Queries expanded to identify all subtypes



A query that explicitly includes the *Concept* IDs of all *subtype descendants* of the *Concept* to be retrieved can be built using one of the following methods:

- A recursive tree-walk following | is a | Relationships - from the selection *Concept* to its *subtypes* and the *subtypes* of its *subtypes*. Each branch of the tree walk ends on reaching a *Concept* with no *subtypes* or a *Concept* that is already in the set of selected *Concepts*.
- Using pre-generated branch number ranges associated with the selection *Concept* and looking up all *Concepts* with branch numbers in those ranges. This could be much faster than a tree-walk if *Concepts* are indexed by branch-number.
- Using a stored list of *subtype Concept* IDs for frequently queried *Concepts*. This would initially be generated in one of the other methods and then reused in various queries. Any stored list would need to be rebuilt after installing each release of *SNOMED CT*.

The resulting *query* may contain a large list of potential *Concept* IDs, but the actual *query* structure is simple. Therefore as long as the database engine does not restrict *query* size, this type of *query* can be run in any environment that support SQL or an SQL-like *query language*.

This technique is likely to be most effective when a large number of candidate record entries need to be examined and when *Concept* selection criteria are relatively narrow. Selecting all diagnoses using this approach would generate a predicate with tens of thousands of *Concept* IDs. Extremely large queries may not perform efficiently or may fail to run in some environments.

8.3.2.2 Subtype tests on each recorded concept



The *Concept* recorded in each candidate record entry can be tested to determine whether it is a *subtype* of the *Concept* to be retrieved. The test can be applied in one of the following ways (see also Testing and traversing *subtype relationships*):

- A recursive tree-walk following | is a | Relationships from the recorded *Concept* to its supertype and the supertypes of its supertypes. Each branch of the tree walk ends on reaching the *Root Concept* or a *Concept* that has already been visited. The test ends with a positive result if the selection *Concept* is encountered during the tree walk. Otherwise when all supertypes have been visited, the test ends with a negative result.
- Optimized *subtype* testing using techniques such as branch numbering and tree-walk enhanced with semantic-type *Identifiers* or *hierarchy* flags.

This technique is likely to be effective when the number of candidate record entries to be examined is relatively small or if the *Concept* selection criteria are broad. Performance is directly dependent on the time taken for each *subtype* test. Therefore, extensive use of this approach may only be feasible by applying one or more of the optimizations discussed in the guide.

8.3.2.3 Use a database with built in hierarchical functionality



Some databases have features which build in hierarchical functionality. These databases may support extensions to SQL that allow a predicate to be specified in a way that implies that the database schema "understands" the *subtype hierarchy*.

Example:

It is possible to envision a statement such as:

WHERE Record.Expression SUBTYPE-OF 414024009

If a database supports this type of predicate, it clearly simplifies the writing of *SNOMED CT* queries. It is also reasonable to assume that functionality of this type, built into a database engine rather than added as an afterthought, will deliver enhanced performance. However, this assumption should be tested as it depends on how appropriate the internal implementation is to *subtype hierarchy* of the size and complexity of *SNOMED CT*.

8.3.2.4 Branch-range indexing of individual records



Branch numbering is an approach to *subtype* testing that could be extended to index record entries. The branch numbers could be used to produce an index of all record entries stored in an application. The technique is as follows:

- Every record entry is indexed using the branch number of the *Concept* stored in that entry;
- The set of branch number ranges associated with the selection *Concept* is then used to *query* the branch number index.

This approach is likely to deliver high performance retrieval but it has a significant drawback. Branch numbers have to be regenerated after each *SNOMED CT release* and the numbering changes each time. Therefore, any indices based on branch numbers must also be rebuilt after each release, and until this rebuild is complete, this method cannot be used for retrieval. The previous set of branch numbers could be used for retrieval during the transition period but this requires a parallel set of branch numbers and branch number ranges.

The likelihood of enhanced retrieval performance should therefore be balanced against the addition of complexity to terminology updates and record maintenance.

8.3.2.5 Retrieval Based on other Relationships



While many queries will use *SNOMED CT's* hierarchical *subtypes* to aggregate data, the attribute *relationships* can also be used. For example, to find all procedure *concepts* that use a laparoscope, search in the *Relationships Table* for *Concepts* with a *relationship* of Using Access Device: Laparoscope. Note that role hierarchies can be used to construct these queries.

8.3.3 Selective retrieval of postcoordinated expressions



The previous section deals with the retrieval of records that contain *precoordinated* representations of *Concepts*. The mechanisms and methods discussed in that section need to be extended to cover *postcoordinated expressions*.

The selective retrieval mechanisms applicable to *postcoordinated expressions* depend on the way in which this data is stored. If data is transformed to generate tables or indices that facilitate retrieval, the form of this derived data determines the type of mechanisms that can be used.

There are two significant factors in the completeness, specificity and performance:

- The structure used for representing *postcoordinated expressions*;
- Whether the information is only stored in the form entered or is also stored in a manner that seeks to facilitate efficient and consistent retrieval of *postcoordinated expressions*.

8.3.3.1 Retrieval from unrestricted relational representations of postcoordinated data



Unrestricted relational representations provide a flexible medium for storage retrieval of *postcoordinated expressions*. A *query* can be specified at any level of detail to examine the primary *Concept* in a statement and any or all of the associated *postcoordinated* qualifications, modifications, or combinations.

However, the number of joins required to specify an appropriate *query* may affect performance.

- Each clinical statement consists of a row in one table joined to a row in a *qualifier* table for each *postcoordinated* refinement. The clinical statement itself may have other structural relations (based on the record structure) and each patient record may consist of hundreds of thousands of statements.

The effect of this will vary according to the power and configuration of the relational database. However, some application developers may seek alternative, more limited representations to improve performance.

8.3.3.2 Retrieval from restricted postcoordinated information



An application may store data in a restricted relational representation, which limits *postcoordination* to a pre-specified set of *qualifiers*. Criteria for selection based on the values of a limited set of *qualifiers* require a minor extension to any of the approaches discussed in the previous section. However, there are two significant points to note:

- When applying criteria to the values of a *qualifier*, any *subtype* of the specified value should be selected. This is similar to the consideration for the primary *Concept*. However, the number of tests to be performed will be more limited because:
 - Typically a *qualifier* value will have relatively few *subtypes*;
 - Only record entries that match on other criteria need to be tested.
- Some of the supported qualifying attributes may also occur in *defining characteristics* of some *Concepts*. A *query* that specifies the presence of a particular *qualifier* must not miss these cases. One way to address this issue is to ensure that when storing or transforming data for retrieval, the value of any *defining characteristics* that are also supported, and qualifying attributes should be copied into the qualifying value field.

8.3.3.3 Retrieval from postcoordinated data stored as parsable text or XML



Parsable text strings or XML elements are not well suited to rapid retrieval from large populations of records. However, optimization is possible by augmenting the stored form with indexes to *Concepts* (e.g. indexing *Concept Identifiers* or range number) or by using an XML-aware database. Without such optimization it may be possible to achieve acceptable performance for retrieval from individual records, documents or messages represented in a structured form using XML.

8.3.3.4 Retrieval of postcoordinated data stored as entered



Where *postcoordinated* data is only stored as entered, retrieval mechanism must do all the hard work of calculating the *equivalence* between statements expressed in different ways. This is possible for a small-scale search (e.g. within a single patient record) but across a large population of records it may be difficult to achieve an acceptable performance.

8.3.3.5 Retrieval from minimized postcoordinated forms



If *postcoordination* is minimized before storage, this allows most of the search process to be concerned with querying or testing *subtype descendants*.

If the *query* needs to specify selection criteria that cannot be expressed by a single *Concept*, further testing is required. Even then, if there are rules for consistently minimizing *postcoordination*, most queries remain easy to construct and apply.

Some complex queries may present more difficulties with this approach but it remains a reasonable option for application developers concerned with minimizing the overhead related to storage and retrieval while delivering reasonable performance and flexibility.

8.3.3.6 Retrieval from maximized precoordinated forms



Maximization of *postcoordination* offers them most flexible approach. Of the three forms suggested:

- The exhaustive form:
 - Simplifies queries since everything that is true about a *Concept* is stated and there is no need to check *subtype* descents;
 - Carries a heavy storage penalty for every record stored;
 - Requires computation of the representation of each *Concept* after every release.

- The long *canonical form* :
 - Allows queries that are relatively simple provided that a mechanism exists for checking *subtype* descents.
 - Although more terse than the exhaustive approach, storing this information for every record stored still has a significant storage overhead.
 - Requires rechecking or re-computing after a release, but this can be done directly from the *release files* by combining the " | is a | relationships in the *Canonical Table* with the other (i.e. not " | is a | ") defining characteristics in the *Relationships Table*.
- The short *canonical form* :
 - Requires slightly more care in construction of queries than the long *canonical form*;
 - Requires slightly less storage than the other maximized forms;
 - Like the long *canonical form*, can be rebuilt directly from the release tables.

8.3.4 Requirements for specific uses of selective retrieval



8.3.4.1 Specifying retrieval requirements



An application should provide a mechanism to allow users to specify retrieval requirements using *SNOMED CT*. This facility should allow queries to be generated that combine *SNOMED CT* specific selection criteria with other health record criteria.

A terminology *browser* that combines text searches and *subtype hierarchy navigation* is likely to be essential for defining *SNOMED CT* specific selection criteria.

Facilities for testing and traversing *subtype relationships* are essential for running most *SNOMED CT* queries that can be run against stored records. Additional functions that take account of the definitions of *concepts* and the refinements in *expressions* are needed to support more sophisticated retrieval.

8.3.4.2 Selective retrieval for reporting and analysis



Population-based retrieval and reporting is usually a task that can be run in the background or scheduled for later execution. Therefore, real - time responses are usually not essential.

The process of analyzing a large number of records may take several minutes or perhaps even hours. If the application spends a little time generating an optimized *query* before starting to access the records, this is acceptable and may shorten overall execution time. Therefore, a technique such as *query expansion* may be appropriate for these tasks.

Users may also have requirements for reports on individual patients or a small group of patients. In some cases there may be an expectation of a real - time response to requests for these reports. If so, the delay while several selection criteria are expanded may be unacceptable. If the same criteria are used many times, storage of the expanded form may be a realistic option. Otherwise, an alternative retrieval technique should be considered.

8.3.4.3 Selective retrieval for decision support



Decision support tools are usually used during a consultation with a patient. Real - time response without significant delay is essential if these tools are to be used regularly and perceived as a help rather than a hindrance. A decision support algorithm may need to selectively retrieve several records to inform a single decision or piece of advice. Many of the retrieval criteria are likely to be quite general. The time taken to expand an apparently simple set of criteria so that they include all appropriate *subtypes* is likely to significantly impair performance. The expanded criteria could be stored in or associated with the protocol. However, the requirement to update these with new *SNOMED CT* releases and whenever the protocol changes add to the maintenance burden.

Since decision support protocols are primarily concerned with the records of an individual patient, it may be feasible to test all candidate records (e.g. all records that fall within a specified date range) to see if any of these are *subtypes* of the selection *Concept* (s).

Other alternatives should also be considered.

8.3.4.4 Decision support tools as authors of data in the record



As well as retrieving *SNOMED CT* encoded information a decision support tool may need to make entries in the record. These entries may arise directly from user interaction with a template or protocol. However, some entries made by decision support tools may record decisions made by or advice given by the tool.

8.3.4.5 Retrieving records encoded with Inactive Concepts



Records that have been encoded using *Concepts* that are no longer *active* can be retrieved by using the Historical Associations Reference Set values (i.e. "same as," "may be a," | replaced by | and "was a") in addition to | is a | *subtype relationships*.

An application should allow users to specify which (if any) of these *Relationships* or Associations should be followed when determining whether to retrieve a record entry.

- A sensible default is to treat duplicate *Concepts* related by | same as | associations and erroneous *Concepts* related by | replaced by | *Relationships* as though they were interchangeable with the related *Concepts*.
- In the case of the ambiguous *Concepts* related by | may be a | associations the solution is less clear-cut. A choice must be made between the importance of completeness, which is best served by including these *Concepts*, and selectivity, which is better served by excluding them.

8.3.4.6 Retrieving and analyzing legacy coded data



SNOMED CT can also be used for generated queries that examine legacy data recorded using *SNOMED International*, Clinical Terms Version 3 or earlier versions of the Read Codes. This can be done by using the approach outlined in [strategies for data migration](#), to generate a query that includes all the *subtypes* of a selection *Concept*. However, the appropriate legacy codes (i.e. The CTV3ID or *SNOMEDID*) are added to the query instead of the *Concept Identifier*.

8.4 Communicating Expressions



SNOMED CT enabled applications must support inbound and outbound communication of information that includes relevant *SNOMED CT* expressions. This section provides an outline of some general issues related to communication of *SNOMED CT* information using standard communication structures.

8.4.1 Representation of *SNOMED CT* information in communications



Various media exist for communicating computer processable data between applications. These include:

- Messages;
- Structured documents;
- Portable storage media (smart cards, memory cards or other similar devices);
- Application interfaces (including COM / CORBA and programmable web interfaces).

A common feature of any method of communication is the need for a formal standard (or de-facto agreed) representation of the communicated information.

To enable full communication of *SNOMED CT* information these agreed standards must allow the communication of *precoordinated* or *postcoordinated* representations.

Some *current* standards do not provide explicit support for *postcoordination*. Examples include:

- *HL7 Version 2.x* messages;
- EDIFACT implementations of European (*CEN*) Prestandards for laboratory communication used in by the UK NHS.

In these cases, it may still be possible to include *postcoordinated* information by agreeing to a syntactic representation that can be used in a single message element.

 **Example:**

Subject to message field length *constraints*, an expression in *compositional grammar* could be included in place of a simple code.

The use of this type of technique is not recommended since it may distort the intended semantics of the message, but also, and more significantly, it requires the recipient to agree to parse the code in a particular way. There is no point sending a parsable text representation of a *postcoordinated Concept* to recipients with no understanding of that form of representation.

More recent standards make specific provision for the support of *postcoordination* in representations of clinical statements. Examples include:

- *HL7 Version 3*, which includes the "*concept description*" (CD) data type which provides unlimited scope for *postcoordinated* modifiers;
- European (*CEN*) Prestandard ENV13606 for Electronic Healthcare Record Communication, which include a component name structure element, which permits *postcoordination*.

Communication of *SNOMED CT* data using explicit structures for *postcoordination* is strongly recommended. However, where local agreements permit, other solutions may be used. This is discussed further in the next section.

8.4.2 Overlaps between *SNOMED CT* and Structural Semantics



Many communication constructs have a built-in, or assumed semantic model.

 **Example:**

Rather than having a single coded expression to represent a procedure the *HL7 Version 3* class Procedure contains the following coded Attributes .

- Code (cd);
- Priority (priority_cd);
- Reason (reason_cd);
- Method (method_cd);
- Procedure_site (procedure_site_cd);
- Approach_site (approach_site_cd).

Similar constructs occur in other *HL7 Version 3* classes (e.g. Observation) and message standards from other sources. However, the *HL7 Version 3* Procedure example shown here is probably the best example of a particular dilemma for those communicating with a message that takes some aspects of semantics to the structural level while leaving others to the coding scheme.

Suppose we want to communicate the following procedure:

- "Emergency removal of foreign body from stomach by incision".

The *HL7* Procedure class would allow this to be communicated in several different ways.

- The first option uses the *postcoordinated SNOMED CT expressions* and leaves the structural Attribute blank;
- The second uses the structural Attribute and does not postcoordinate the information in the expression;
- The third duplicates the same information both in the structure and in the *postcoordinated expression*.

These options represent the main type of approach to overlaps. However, in practice the structural model may permit similar information to be recorded in more than one way and *SNOMED CT expressions* also offer alternatives depending on the extent of normalization of the representation.

The main point is to stress the potential for confusion even when using the same communication structure and the same terminology. This is not a specific problem for *SNOMED CT* or for a particular message design. Any combination of structural and terminological semantics is susceptible to this issue. Since effective communication of information requires both structure and terminology the challenge is to define an interface between structural and semantic models so that they form part of a common *model of meaning*.

8.4.3 Using Reference Sets to represent allowable value sets



Standard message specifications and communication agreements with particular user communities often specify restricted lists of codes that can be used in particular message elements:

- Examples of this include the *HL7* idea of "vocabulary domains" containing "value sets" specified for use either in a general or specific context in a message element;
- The *UK NHS* specification for laboratory report messages, which refers to a "bounded list" of *Read Codes* that are to be used in particular fields of the message.

It is inevitable that a broad terminology such as *SNOMED CT* needs to be restricted in this way. A message element intended for representation of a | requested radiology investigation | must clearly contain a *Concept Identifier* that represents a radiology procedure. The limitations may go further than this. The list of procedures that can be requested may be restricted by local convention or regulation.

A *Concept Reference Set* can be used to represent *value sets* that are permitted in a particular type of message or within a particular user community. This facilitates use of a general-purpose *SNOMED CT* enabled *Terminology services* to populate and validate the coded elements of messages.

8.4.4 SNOMED Clinical Terms and HL7



HL7 develops standards for the exchange of health related data. Most new *HL7* standards developed over the last few years have been based on *HL7 Version 3*. The key features of *HL7 Version 3* include:

- A *Reference Information Model* (RIM):
 - This provides a framework for the structure of communicated information.
- A formal development method:
 - This described the various steps to turn a set of requirements into appropriate models and specifications that support communication of the necessary information.
- Separation between logical models and implementation technologies:
 - Many implementations use XML but other technical approaches can be applied to implement the same models.
- Use of external codes systems and terminologies to represent concepts:
 - Model specifications include coding constraints expressed in abstract terms as Concept Domains which are implemented as *Value Sets* drawn from one or more code systems.

Many *HL7 Version 3* models use *SNOMED CT* and this has led to growing demand for guidance on consistent patterns of use. In 2004, the *HL7 Vocabulary Technical Committee* launched the *TermInfo Project* to address this requirement. The project was initially conceived as having two work packages:

1. Specification of a general approach to resolving issues related to the interface between *HL7* information models and terminologies or code systems;
2. A guide to use of *SNOMED Clinical Terms* in *HL7 Version 3* communication standards.

The *SNOMED CT* specific package was actively supported by *SNOMED International* as part of a charter agreement with *HL7*. After several rounds of revision and review, the 'Guide to Use of *SNOMED CT* in *HL7 Version 3*' was accepted by *HL7* as a *Draft Standard for Trial Use* (DSTU).

The guide itself contains both normative and informative sections. The normative sections cover:

- Guidance on dealing with specific overlaps between RIM and *SNOMED CT* semantics and recommendations for use of *SNOMED CT* in relevant *Attributes* of various RIM classes;
- Constraints on *SNOMED CT concepts* applicable to relevant *Attributes* in each of the major classes in the Clinical Statement pattern.

The informative sections cover:

- Examples and patterns for representing common clinical statements;
- A general discussion of the potential overlaps between an information model and a *terminology model* and the pros and cons of various possible approaches to their management;
- References to relevant documents and known open issues.

Following adoption the *TermInfo* group have encouraged in-use testing and have elicited and addressed comments on the 'Guide to Use of *SNOMED CT* in *HL7 Version 3*'. Implementer feedback has further contributed to growing understanding of the issues and resulted in refinement of the guidance. Many of the recommendations included in the 'Guide to Use of *SNOMED CT* in *HL7 Version 3*' have been incorporated into domain specific *HL7 Standards* and national implementations.

One of the conclusions from the *TermInfo* group is a recognition that terminology and information models are co-dependent. They need to evolve collaboratively to meet requirements for unambiguous processable representations of information. Work with other Information Models (including *EN13606* and *openEHR*) indicates that the issues raised by *TermInfo* are not specific to *SNOMED CT* and *HL7* (*Representing clinical information using SNOMED Clinical Terms with different structural information models*). Harmonization efforts involving the *IHTSDO* and *HL7* and other standards bodies continue to address these issues.

As part of the ongoing harmonization work the copyright of the '[Guide to Use of SNOMED CT in HL7 Version 3](#)' is jointly held by *HL7* and *IHTSDO*. In line with *HL7* policies, the document expired as a formal *HL7 DSTU* after two years but it is included here in full as it continues to serve two *Roles*:

- A *Description* of the challenges of integrating an expressive terminology, such as *SNOMED CT*, with a rich information model;
- A pragmatic interim approach to these challenges, which allows for and anticipates the evolution of a more integrated solution.

Chapter

9

9 Change Management Guide



This part of the guide addresses requirements that arise from changes to the content, structure and use of the terminology.

The first significant change management challenge relates to *migration* from other coding schemes or from a less structured electronic record system. Decisions must be made about retaining or converting records, queries and protocols originally created using a terminology other than *SNOMED CT*.

- The initial content of this section focused on *migration* from *SNOMED RT*, *SNOMED International*, *Clinical Terms Version 3* and the *Read Codes*. Some of the general points also apply to *migration* from other code systems.

Each release of *SNOMED CT* introduces some changes to content. Many of these changes are additions to breadth and depth of coverage. There may also be corrections to *concept* definitions and enhance the expressivity of the *Concept Model*.

- These changes are an essential characteristic of an evolving clinical terminology that seeks to support changing requirements. However, significant changes need to be evaluated and managed to assess and adjust for any effects they may have on the data entry and retrieval and validity and comparability of data from different sources.

Occasionally there may also be additional technical artifacts or specifications developed to meet emerging requirements.

- Additional material related to updating to support *Release Format 2* has also been included and provides an example of guidance on technical changes.

As systems evolve and as the content and structure of *SNOMED CT* are enhanced there is a continuing requirement to manage changes smoothly and without loss of information or functionality.

- As experience grows, this guide will be further developed to address a broader range of change management issues.

9.1 Managing Content Changes in SNOMED CT



This section of the guide addresses potential issues that may affect implementations when new releases of *SNOMED CT* contain changes to content.

The likely impact of four general types of change are considered.

- Content additions.
- Content inactivation (e.g. inactivation of a *Concept*).
- Changes to *relationships* between *concepts*
- Changes to the *Concept Model* leading to systematic changes to a range of components.

The impact of these changes is assessed in *terms* of the main *record service* functions

- Data entry (e.g. potential requirements to change to data entry protocols).

- Data storage (e.g. whether pre-existing data needs to be migrated or processed in a particular way to ensure consistency).
- Data retrieval (e.g. whether existing queries are likely to need revising to take account of the changes).
- Communications (e.g. whether communication specification are likely to be affected and in particular potential issues from cross-version communications).

Following discussion of these general considerations, the remainder of this section holds specific advice related to any content changes in a release which are expected to require attention from implementers.

9.1.1 Changes and historical notes



9.1.1.1 EPISODICITY no longer modeled in active content



| EPISODICITY | originated in the *National Health Service Clinical Terms Version 3* where it was used not to specify the first episode of a disease for a patient but rather, the first time a patient presented to their general practitioner (GP) for a particular disorder. A first episode of asthma was not intended to represent the first time a patient had asthma, but rather the first time a patient presented to their GP with asthma. | EPISODICITY | has been removed from existing *concepts* and is no longer used in *precoordinated* definitions. It can still be used in *postcoordination* as a *qualifier*.

9.1.1.2 ONSET and COURSE retired



In earlier releases, there were two attributes named | ONSET | and | COURSE |. These were *retired* because they could not be used reproducibly. While | ONSET | was intended to specify the rapidity of onset or the temporal pattern of presentation for a given condition, it was easily confused with the attribute | COURSE | used to represent the duration of a condition. There was not consistent agreement between observers making this distinction.

9.1.1.3 Dose form values moved



The *concept* 105904009 | Type of drug preparation (product) | and its *subtypes* were moved to the *Qualifier value hierarchy* as of the July 2007 release. 105904009 | Type of drug preparation (qualifier value) | better represents these *concepts* because they are not products.

9.1.1.4 Renaming the context/situation hierarchy



The *hierarchy* named 243796009 | situation with explicit context (situation) | was called | context-dependent category | until the July 2006 release. The *hierarchy* was renamed to better describe the meanings in this *hierarchy*.

9.1.1.5 Domain change for measurement/evaluation attributes



In releases prior to July 2009, six *attributes* were approved for use for | measurement procedure | only. For the July 2009 release, the *domain* for these *attributes* was expanded to | evaluation procedure |. See [Measurement procedures and laboratory procedures](#) for a definition and full discussion of | evaluation procedure | and | measurement procedure |.

9.1.1.6 Move of findings to events



In January 2006, a number of *concepts* from the | Clinical finding | *hierarchy* were moved to the | Event hierarchy. The attributes used to define those *concepts* when they were *descendants* of | Clinical finding | were retained after the *concepts* were moved to the | Event hierarchy. Additional editorial policies for the use of attributes in the | Event hierarchy have yet to be established.

9.2 Managing Technical Changes in SNOMED CT



This section of the guide addresses potential issues that may affect implementations when new or revised *SNOMED CT* technical specifications are planned or released.

Each subsection deals with a specific proposed or actual change.

9.2.1 Release Format 2 Update Guide



9.2.1.1 Introduction

9.2.1.1.1 Purpose



The purpose of *RF2* is to provide a format that is flexible, unambiguous and useful. Its primary aim is to strengthen *SNOMED CT* by providing a format that is simple and stable, while enabling innovation through adaptations to cater for changing requirements.

This specification was developed by harmonizing proposals reviewed by the *IHTSDO Enhanced Release Format Project Group*, including:

- The Enhanced *Release Format Specification* (International Health Terminology Standards Development Organisation. *SNOMED Clinical Terms ® Enhanced Release Format Proposed Specification* , 21 June 2007).
- The Reference Set Specification (International Health Terminology Standards Development Organisation. *SNOMED Clinical Terms ® Reference Sets - Proposed Specification* , 31 July 2007).
- The Alternate *Release Format* proposed by NEHTA in coordination with their Australian Affiliates.

9.2.1.1.2 Who should read this guide?



The intended audience for this guide includes technical professionals who are involved in the development and/or implementation of healthcare information systems that use *SNOMED CT*.

For detailed technical guidance on the existing *Release Format*, please consult the *SNOMED CT Technical Reference Guide* (TRG) and *SNOMED CT Technical Implementation Guide* (TIG), as well as other applicable technical documentation described in the Associated Documentation table.

For technical guidance on using *Release Format 2*, please consult the "*SNOMED CT Release Format 2 - Reference Set Specifications*" and the "*SNOMED CT Release Format 2 - Data Structures Specification*" documents on the Collaborative site.

9.2.1.1.3 Associated Quality Measures



Although the definition of quality measures to monitor the implementation of this standard do not fall under the scope of this guide, they will be covered by the documentation covering the QA and *Release* process for the *Workbench*.

9.2.1.1.4 Summary of Changes



The *RF2* introduces a number of new *concepts* and capabilities. These are summarized below, and described in more detail later in this guide:

- Addition of an *Identifier file* to allow components to be identified by an arbitrary number of *Identifiers* from an arbitrary number of *Identifier* schemes;
- Addition of a module *Identifier* field to all components, enabling the source module in which each component is maintained to be identified, facilitating configuration management;
- Modified handling of the *language* and *dialect* properties of *descriptions*, for reduced complexity with increased utility;
- Introduction of *concept* enumerations making enumerations within *SNOMED CT* more easily extensible, self contained within the terminology (not dependent upon external documentation) and easily compatible across multiple *languages*;

- Addition of a *description logic* modifier *concept* enumeration to the *Relationship file* to represent different *Description Logic relationship types*, for example - some, all, all-some, not-some etc.

A general extensibility design pattern has also been introduced, which allows specification of a number of *Reference Set* formats, to meet different use cases. In *RF2*, *reference sets*:

- Result from the combination of generic *Reference Set* data structures, a design pattern and the application of domain *constraints* according to documented implementation guidelines;
- Use a machine readable model (called a *Reference Set descriptor*) that defines the extended information pertinent to a specific *Reference Set*;
- Make use of *concept* enumerations for representing optional information to enable machine-readability and increased extensibility;
- Apply the same history tracking and naming conventions as used elsewhere in *RF2*.

The *RF2* enhancements all contribute to greater flexibility and more explicit and comprehensive version control than *RF1*, and additionally introduce new features for configuration management. As a result, *RF2* is expected to accommodate evolving collaborative requirements without a need for further fundamental change in the foreseeable future. Since change to the *Release Format* causes difficulty and incurs cost to content developers, implementers and release centers alike, the *RF2* design is expected to result in long term savings as well as improvement in product functionality and quality.

9.2.1.1.5 Timescales for change



It should be noted that there is a difference between the release schedule of *RF1* / *RF2* in official IHSTDO-supported *International Releases*, and the release schedule of *RF1* / *RF2* in Member NRC releases. It is entirely possible that *RF1* will have a longer lifespan in Member NRC releases than in *IHTSDO International Releases*.

Actual timescales for *migration* of the *International release* to *RF2* are provided under separate notices, and have not been included in this guide as they are likely to follow a different review cycle.

9.2.1.2 Principles used in the design of RF2



The following principles were used to guide modifications made to the *Release Format*:

- Consistent history representation across all components and across all artifacts deemed in scope of the *Release Format*.
- Consistent identification of all components throughout their lifecycle and clear identification of all other artifacts in scope of the *Release Format*.
- Consistent representation of allowable values for component characteristics.
- Consistent means of extending component data structures to meet future requirements without modification to the existing table structures.
- Consistent non-centralized means of loosely coupled *Identifier* assignment for components and component characteristics.
- Consistent means of representing localizations and translations for all components.
- The data structures should assist implementers to consistently implement *SNOMED CT*. Component data structures ideally should not have to change to accommodate changes in editorial policies.
- Ideally component data structures should be simple, generic and flexible.
- Ideally component data structures should be self-contained, removing dependence on external artifacts .
- Dependencies between components should be explicitly stated and machine-readable. For example, it should be possible to express that a *reference set* released as part of an *extension* is dependent upon version X of the Acme *Extension* and version Y of the *SNOMED CT International Edition*.
- There must be a consistent means of identifying modules and their versions --including the *SNOMED CT International Release* itself.
- The *Release Format* should minimize the total effort of meeting requirements where possible by reuse of existing data structures.
- Metadata should be machine-readable.

- Component data structures that enable software reuse are preferred over data structures that require special development of parsers.
- It should be possible to produce from a *Release Format* an instance of that release in the immediately prior *Release Format*.
- Specifications should be based on requirements derived from use cases that describe the scope and environment of their intended use.
- *IHTSDO* specifications should provide a common global foundation that permits the development and maintenance of *SNOMED CT enabled applications* that are interoperable across national and organizational boundaries.
- Changes to *IHTSDO* specifications should only be made if the impact on implementation is considered to be proportionate to benefit. Such changes should be recorded.
- Changes to *IHTSDO* specifications should be evolutionary and should deliver incremental benefits to implementers with a minimum of disruption and re-engineering.
- The *SNOMED CT release format* and associated guidance should facilitate a consistent implementation for known use cases.
- The specifications that support implementation of use cases should be done in a way that doesn't limit the ability to realize other use cases within the scope of *SNOMED CT*.
- The *Release Format* is intended to be a distribution format and is not designed to be an implementation format.
- The *Release Format* should be designed to be consumed efficiently.

9.2.1.3 Rationale for moving from RF1 to RF2

9.2.1.3.1 Overview of Release Format 1

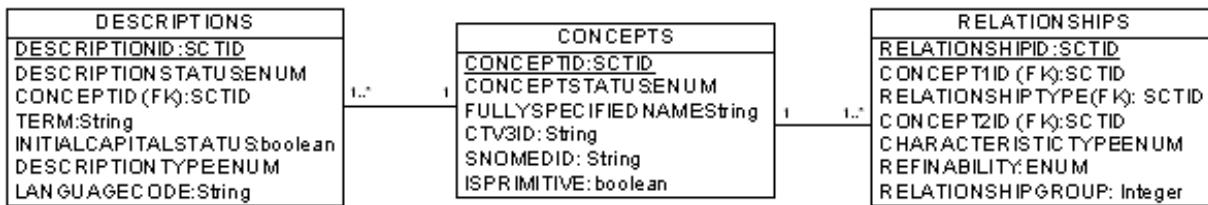


The *current SNOMED CT release format* has been in use since January 2002.

During this period the generic and reusable aspects of the existing release structure have been a considerable strength.

Despite this success, there are a number of commonly accepted inconsistencies and limitations in the *current SNOMED CT distribution format*. This section gives a brief overview of the *current SNOMED CT distribution format*, and describes these limitations. For more details see the [Release Format 1 - Detailed specification](#).

The current *RF1* format is summarized in [Figure 125](#).



Each *SNOMED CT concept* is held as a single row in the *CONCEPTS* file. Each *concept* may have one or more *descriptions* associated with it. Each *description* is held in a single row in the *Descriptions* file. Each *relationship*, from a source *concept* to a destination *concept*, is held as a single row in the *Relationships* file. The type of each *relationship* is defined by reference to a linkage *concept*, also held within the *CONCEPTS* file.

Separate file structures are also released to provide support for *Cross-Maps* to other terminologies and coding systems, history tracking of *components* and *descriptions*, subsets of *SNOMED CT concepts* and other uses.

9.2.1.3.2 Drawbacks of Release Format 1

Inconsistencies and limitations in the *current Release Format* have led to a desire for a new *Release Format*. The following list briefly outlines these issues:



- Implicit semantics that must be inferred from external documentation that is not tightly coupled to changes in the terminology itself. Changes in the interpretation of data fields are not represented in the history of the data fields themselves.
- Pervasive use of *integer* enumerations within data fields, rather than using the self-referential means for representing symbolic constants provided for by the *SNOMED CT* terminology itself.
- No consistent and clearly defined mechanism for release centers , developers, implementers, and end users to extend the *RF1* data structures to meet unique and/or common needs not already provided for by the specifications and content of the *SNOMED CT International Release*.
- Inconsistent and unnecessarily complex data structures.
- Field overloading ""where one column represents multiple attributes (i.e. state and *reason* for inactivation).
- Inadequate Separation of Concerns, where data representation and data usage are often conflated, resulting in a difficulty in supporting software reuse and system evolution over time.
- Inconsistent and incomplete representation of terminology history resulting in a terminology that does not meet basic principles of configuration management and control.
- Inconsistent use of both enumerated values and *concepts* to enumerate values.
- Inconsistent naming and field ordering.
- *Term* length limited to 255 bytes and to plain text format.

Release Format 2 aims to address these issues.

9.2.1.3.3 Overview of Release Format 2



Release Format 2 consists of four primary files or tables. As in the *current SNOMED CT* format, all files:

- are tab delimited text files;
- are *UTF-8* encoded;
- contain a column header row;
- use DOS style line termination (i.e., all lines including the final line are terminated with a carriage return character followed by a line feed character).

The core table structure of *RF2* is similar to that of *Release Format 1*, although the fields within each of the core files are different. The core files within *RF2* consist of a *Concept file*, a *Description file*, and a *Relationship file*.

Each *SNOMED CT concept* is held as a single row in the *Concept file*. Each *concept* may have one or more *descriptions* associated with it. Each *description* is held in a single row in the *Description file*. Each *relationship*, from a source *concept* to a destination *concept*, is held as a single row in the *Relationship file*. The type of each *relationship* is defined by reference to a linkage *concept*, also held within the *Concept file*.

In addition to these files, an *Identifier file* has been added. This file holds one alternate *component Identifier* per row. Each alternate *Identifier* belongs to a particular *Identifier* scheme, and holds that scheme's *Identifier* for the *SNOMED CT Component* that it references. Within a scheme, each *Identifier* uniquely identifies a single *SNOMED CT component*.

The purpose of *RF2* is to provide a format that is flexible, adaptable, consistent, unambiguous and above all useful. Its primary aim is to strengthen *SNOMED CT* by providing a format that is simple yet flexible and powerful, allowing the format to remain constant, while allowing innovation and adaptation to changing requirements.

9.2.1.3.4 Backward compatibility



The proposed *RF2 Release Format* is backward compatible with the previous *SNOMED CT Release Format (RF1)*, in the sense that all information contained within the *current Release Format* is represented, and legacy file formats can be derived from the new *Release Formats*. However, the *RF2* format contains functionality which is not supportable in the previous *Release Format*.

In order to achieve backward compatibility, the *RF2* may be transformed to create the previous distribution format. Additionally, *International releases* will be made available in both formats for a limited number of consecutive release cycles, for convenience. It is expected that *National Release Centers* will follow the same

approach, and also release in dual format for a number of release cycles, unless there are specific *reasons* not to.

9.2.1.4 Details of Key Changes



The following subsections discuss details of the key changes between *RF2* and the previous *Release Format*.

9.2.1.4.1 Addition of effectiveTime and active fields



The *effectiveTime* and *active* fields enable the use of a "log style" append-only data model to track all changes to each component for full traceability. Historic data will be supplied in the *RF2 release files*, dating back to the first release in the *current* format in 2002.

Once released, a row in any of the *RF2 release files* will remain unchanged through future releases. In order to change certain properties of a *current* component (and, therefore, to create a new version of it), a new row must be added to the applicable file, containing the updated data. The *active* field in the newly added row is set to true and the timestamp in the *effectiveTime* field indicates the point in time at which the new version comes into effect.

By contrast, where editorial policy does not allow a particular property of a component to be changed whilst keeping the same *Identifier*, the component as a whole is inactivated by adding a new row containing the same data as the final valid version of the component, but with the *active* field set to false and the timestamp in the *effectiveTime* field indicating the nominal release date at which the final version ceased to be valid.

It is thus possible to see both the *current* values and any historical values of a component at any point in time.

9.2.1.4.2 Active field



As mentioned above, each file contains a *Boolean active* field, used to indicate whether, after the point in time specified in the *effectiveTime* field, the version of the component expressed in the row is *active* or *inactive*.

This field replaces the *status* field with a simple binary state. In the previous *Release Format*, this field was overloaded to enumerate both whether the *concept* was *active*, why it was inactivated, and whether it was about to change (or had changed) authority.

The additional information encoded in *RF1's status* enumeration is represented in *RF2* using the following *reference sets*:

- *Concept* inactivation indicator;
- *Description* inactivation indicator;
- *Relationship* inactivation indicator.

These three *reference sets* conform to the *Attribute Value reference set* pattern, and are further described in the "*SNOMED CT Release Format 2 - Reference Set Specifications*" document.

9.2.1.4.3 History tables



History tracking in *RF2's main files* uses a log-style, append-only data model. Therefore, the separate *ComponentHistory* file that formed part of the original *Release Format* is no longer required with *RF2*.

The associations between *inactive* and *active Concepts* that are currently supported by *Historical Relationship* types (e.g. | SAME AS |, "REPLACED BY") will continue to be supported. References held in the *References table* from an *inactive component* to other equivalent or related *components* that were *current* in the *Release Version* in which that *component* was inactivated will also continue to be supported. However, both of these associations have now moved from the *Relationships* file and the *References* file to one of the following |*Historical association*| *reference sets*.

Table 276: RF1 to RF2 History Field Mappings

RF1 source	RF2 Historical association reference set
MAYBE A (in <i>Relationships table</i>)	POSSIBLY EQUIVALENT TO association <i>reference set</i>
Refers to ('7' in <i>References table</i>)	REFERS TO concept association <i>reference set</i>
Similar to ('3' in <i>References table</i>)	SIMILAR TO association <i>reference set</i>
MOVED FROM (in <i>Relationships table</i>) Moved from ('6' in <i>References table</i>)	MOVED FROM association <i>reference set</i>
MOVED TO (in <i>Relationships table</i>) Moved to ('5' in <i>References table</i>)	MOVED TO association <i>reference set</i>
Alternative ('4' in <i>References table</i>)	ALTERNATIVE association <i>reference set</i>
WAS A (in <i>Relationships table</i>)	WAS A association <i>reference set</i>
REPLACED BY (in <i>Relationships table</i>); and Replaced by ('1' in <i>References table</i>)	REPLACED BY association <i>reference set</i>
SAME AS (in <i>Relationships table</i>) Duplicated by ('2' in <i>References table</i>)	SAME AS association <i>reference set</i>

These *reference sets* all conform to the Association *reference set* pattern, and are further described in the "SNOMED CT Release Format 2 - Reference Set Specifications" document.

9.2.1.4.4 Field naming



Lower camel case has been used for field names in distribution file headers and in documentation that describes these files. File names will use upper camel case (starting with a capital letter). File names have also been altered to use a singular not plural form.

An example of upper Camel Case is ThisIsUpperCamelCase. An example of Lower Camel Case is thisIsLowerCamelCase.

9.2.1.4.5 Field Ordering



Records in the *Concept*, *Description*, *Relationship* and *Reference Set* member files each start with the following four fields:

- *id*;
- *effectiveTime*
- *active*
- *moduleId*

The four fields have the following meanings:

- The *id* field provides a unique *Identifier* for the component described by the record;
- The *effectiveTime* gives the nominal release date at which this version of the component came into effect;
- The *active* flag states whether the components *active* or *inactive*;

- The *moduleId* identifies the source module in which the component is maintained.

The *Identifier file* does not follow the same format, as it works in a slightly different way to the other files, and is described in more detail in the "SNOMED CT Release Format 2 - Data Structures Specification" document.

9.2.1.4.6 Concept enumerations



Concept enumerations have been used across *RF2* to replace *integer* enumerations that can only be understood by referencing external documentation. For example, in *RF1*, a *concept status* value of '4' indicates *concepts* that are *inactive* because they are ambiguous. In *RF2*, *concept* enumeration simply uses *concepts* in a metadata *hierarchy* to represent the enumerated *value set* rather than using arbitrary *integer* values directly. Using *concepts* to represent the enumerated values has the following advantages:

- The terminology is self contained, removing the requirement for external documents to explain the meaning of enumerated values;
- Full *language* handling capabilities are available for the enumerated values' representation, useful for standardized multi-lingual representation, and translation support;
- Machine readable model constructs can be used to further describe and enrich the enumerated values.

The following fields have been converted to *concept* enumerations:

Table 277: RF1 to RF2 enumerated field changes

File	Existing <i>RF1</i> field name	<i>RF2</i> field name
<i>Concept</i>	<i>IsPrimitive</i>	<i>definitionStatId</i>
<i>Description</i>	<i>DescriptionType</i>	<i>typeId</i>
<i>Description</i>	<i>InitialCapitalStatus</i>	<i>caseSignificanceId</i>
<i>Relationship</i>	<i>CharacteristicType</i>	<i>characteristicTypeId</i>

Care should be taken not to confuse *Concept* Enumerations with the term "enumeration" as used in representational formats. A *Concept* Enumeration is a *concept* whose immediate *children* represent possible values in a range. Each possible value is represented by a single *child concept*, whose *preferred term* may be used, for example, to enable selection from a pick-list of one or more values from the range.

Mappings from *RF1* values to *RF2* *concept* enumerations are given below:

Table 278: RF1 to RF2 enumerated value mappings

<i>RF2</i> field name	<i>RF1</i> value	<i>RF2</i> value
<i>definitionStatId</i>	0	Defined
	1	Primitive
<i>typeId</i>	(no specified value)	Definition
	3	Fully Specified Name
	0, 1 or 2	Synonym

RF2 field name	RF1 value	RF2 value
caseSignificanceId	0	Initial character case insensitive
	1	Case sensitive
	(no specified value)	Case insensitive
characteristicTypeId	3	Additional relationship
	0	Inferred relationship
	0	Stated relationship
	1	Qualifying relationship
	2	(no specified value) - now modeled through the <i>inactive</i> association reference set.

9.2.1.4.7 Reference Set Data Structures

9.2.1.4.7.1 Overview of Reference Sets



Reference Set data structures provide the foundation pieces for RF2's generic extensibility mechanism. These building blocks provide a common foundation for extension builders to extend SNOMED CT, and provide RF2 with the capability to grow with the IHTSDO's requirements over time.

Conventions applied to the RF2 files such as field naming, field ordering and history tracking have also been applied to the Reference Set specification. This has been done to provide consistency across all components in the Release Format.

Generic data structures for Reference Sets have been used to create a simple core structure that can be extended to meet a variety of requirements, rather than a complex and inextensible structure that can only be used in a finite and constrained number of ways to enforce editorial policy. This stems directly from a desire to decrease impact on the SNOMED CT community by being able to meet future requirements without having to alter the underlying data structures.

Using these generic structures, it is possible to extend the data stored within the main files of SNOMED CT to satisfy new use cases without altering the primary structure itself. Containing this extended information in externalised structures such as Reference Sets also enables terminology consumers to opt in or out of the content without burdening the primary files with the content. This prevents users from having to download all content and filter out what they don't want, and instead allows them to import the extension content should it be desired.

Reference Sets allow the SNOMED CT core data structures to be extended, allowing existing components to be grouped together into a set, each tagged with a number of additional fields. Each of these additional fields may either be another SNOMED CT component, a string or an integer. Reference set descriptors are also introduced, providing a way to identify the format and purpose of each additional field in a machine readable way. Examples of reference set data structures are provided in the "SNOMED CT Release Format 2 - Reference Set Specifications" document.

9.2.1.4.7.2 RF1 Subsets Representation



The *RF1 Subset* mechanism consists of two tables: a *Subsets table* and a *Subset Members* table. Each row in the *Subsets table* describes a *Subset* and characteristics of that *Subset*, as described in the table below.

Table 279: Subsets Table

Field	Description
<i>SubsetId</i>	The unique <i>SNOMED CT Identifier</i> for this <i>Subset</i>
<i>SubsetOriginalId</i>	The unique <i>SNOMED CT Identifier</i> for the original <i>Subset</i> of which this <i>Subset</i> is a version.
<i>SubsetVersion</i>	An <i>integer</i> incremented for each revised release of a <i>Subset</i>
<i>SubsetName</i>	A name that describes the purpose or usage of this <i>Subset</i> .
<i>SubsetType</i>	Indicates the nature of the <i>Subset</i> and the type of <i>SNOMED CT Component</i> that may be a member of the <i>Subset</i> .
<i>LanguageCode</i>	Identifies the <i>Language</i> and optionally the <i>Dialect</i> to which the <i>Subset</i> applies (only used for <i>description</i> -based subsets: <i>Language</i> , <i>Realm Description</i> , and <i>Realm Concept</i>).
<i>RealmId</i>	Identifies the <i>Realm</i> to which the <i>Subset</i> applies.
<i>ContextId</i>	May identify the <i>Context Domain</i> to which the <i>Subset</i> applies

Each row in the *Subset Members* table sets the *status* of a member of an identified *Subset*.

Table 280: Subset Members Table

Field	Description
<i>SubsetId</i>	The unique <i>SNOMED CT Identifier</i> for this <i>Subset</i>
<i>MemberId</i>	The <i>SNOMED CT Identifier</i> of this <i>Subset Member</i> . This may be a <i>Concept Identifier</i> , <i>Description Identifier</i> or <i>RelationshipId</i> .
<i>MemberStatus</i>	An <i>integer</i> specifying the <i>status</i> , <i>type</i> or <i>order</i> of this member.
<i>LinkId</i>	Valid for <i>Navigation</i> and <i>Duplicate Terms</i> <i>Subsets</i> only. For <i>Navigation Subsets</i> it is the <i>SNOMED CT Identifier</i> for a <i>Concept</i> that is a <i>Navigation child</i> of the <i>Subset Member</i> . For <i>Duplicate Terms Subsets</i> it is the <i>SNOMED CT Identifier</i> for the highest priority <i>Descriptions</i> having the <i>Duplicate Term</i> .

Some *Subsets* and their members are generated automatically from an XML definition file.

9.2.1.4.7.3 Representing Subsets as Reference Sets



An existing *RF1 Subset* may be represented as an *RF2 Reference Set* in the following way:

A *concept* should be created in the *| Reference Set |* metadata *hierarchy*, using information in the *Subset* table record. A *Descriptor* for the *Reference Set* should also be set up using information in the *Subset* table record. Then, one *Reference Set* member record should be created for each *Subset Member* table record.

The way in which the *subsets* are represented in *RF2* depends on the *SubsetType* value, as follows:

Table 281: Representing Subsets as Reference Sets

<i>SubsetType</i> value	<i>Description</i>	<i>Mapping to RF2</i>
1	<i>Language</i>	<i>Language type Reference Set</i>
2	<i>Realm Concept</i>	<i>Ordered type Reference Set</i>
3	<i>Realm Description</i>	<i>Language type Reference Set</i>
4	<i>Realm Relationship</i>	<i>Ordered type Reference Set</i>
5	<i>Context Concept</i>	<i>Ordered type Reference Set</i>
6	<i>Context Description</i>	<i>Language type Reference Set</i>
7	<i>Navigation</i>	<i>Ordered type Reference Set.</i>
8	<i>Duplicate terms</i>	<i>Ordered type Reference Set</i>

9.2.1.4.7.4 Representing Subsets as Ordered type Reference Sets



| *Ordered type | Reference Sets* can be set up as follows:

First, set up a new *concept* for the *Reference Set* in the *|Ordered type|* metadata *hierarchy*. The position in the *hierarchy* should be given by the *RealmId* and *ContextId* fields in the *Subset* record, as follows:

SNOMED CT Model component

Foundation metadata concept

Reference set

Ordered type

RealmId

ContextId

If either the *RealmId* field or the *ContextId* fields are "0", "1", blank or null in the *Subset* record, then that level should not be set up in the metadata *hierarchy*. If a *concept* already exists under *|Ordered type|* with a matching *RealmId* and *ContextId*, then the new *Reference Set* should be set up in that position (as opposed to creating two *|Ordered type| children* with duplicate names).

First, the *concept* describing the *Reference Set* should be created with the following values:

Table 282: Reference Set Concept

Field	Data type	Set to
id	SCTID	A unique id in your <i>namespace</i> .
effectiveTime	Time	The nominal date of release for your <i>Reference Set</i> . If a full state valid representation of a <i>subset's</i> history is required, then each previous release of the <i>Subset</i> files must be processed in turn (by identifying <i>Subset</i> records with a matching <i>SubsetOriginalId</i> , in their <i>SubsetVersion order</i>), and each amended version must be applied to the <i>reference set</i> by appending rows in the usual fashion. The effectiveTime of each applied change should be set to the date that each version of the <i>Subset</i> was released.
active	Boolean	'1'
moduleId	SCTID	The module <i>Identifier</i> for your authoring organization .

Then, add up two *Descriptions* for the FSN and the *Preferred Term* of the *concept*:

Table 283: Reference Set Descriptions

Field	Data type	Set to
id	SCTID	A unique id in your <i>namespace</i> .
effectiveTime	Time	The nominal date of release for your <i>Reference Set</i> .
active	Boolean	'1'
moduleId	SCTID	The module <i>Identifier</i> for your authoring organization .
conceptId	SCTID	The <i>Identifier</i> of the <i>concept</i> describing the <i>Reference Set</i> that you've just added.
languageCode	String	The <i>language</i> of the <i>Description</i> . This field should be set to the <i>language</i> that the <i>Subset</i> was defined in, for example - 'en' for English.
typeId	SCTID	Create two <i>Description</i> records, one for each of the following types: Fully specified name , <i>Synonym</i> .
term	String	The <i>term</i> for the <i>Synonym</i> should be set to the <i>SubsetName</i> field in the <i>Subset</i> record. The <i>term</i> for the FSN should be set to the same, but appended with "reference set (foundation metadata concept)".

Finally, add one *Reference Set* member record for each record in the *Subset Members* table for the *Subset*.

Table 284: Converting a Priority Subset to an Ordered Reference Set

Field	Data type	How to populate
id	UUID	A new unique <i>Identifier</i>
effectiveTime	Time	The nominal date on which this release was made.
active	Boolean	'1'
moduleId	SCTID	Set to the <i>moduleId</i> of the authoring organization .
refsetId	SCTID	A reference to the <i>concept</i> describing the <i>Reference Set</i> that you've just created.
referencedComponentId	SCTID	Set to <i>MemberId</i> in the <i>Subset Members</i> table record.
order	Integer	Set to <i>MemberStatus</i> in the <i>Subset Members</i> table record.

👉 **Note:** Although a *Navigation Subset* can be represented in an *|Ordered type| reference set* as described above, the values of the *linkedTo* field would then have a different meaning, referencing a *child concept* instead of grouping *components* together.

A Descriptor can also be set up for the *Reference Set* if required, as follows:

Table 285: -

refsetId	referencedComponentId	attributeDescription	attributeType	attributeOrder
Reference set descriptor	Concept describing refset	Referenced component	component type	0
Reference set descriptor	Concept describing refset	Order	Unsigned integer	1
Reference set descriptor	Concept describing refset	Linked to	component type	2

Where *Concept describing refset* is the *Concept* that you've just set up to describe the *Reference Set*. The *| Order |* and *|Linked to| concepts* that describe each additional *Attribute* in the *Reference Set* can also be replaced by more descriptive ones if required. To do this, create the new *concepts* describing the additional fields under the *| Reference set Attribute | metadata hierarchy*.

9.2.1.4.7.5 Representing Subsets as Language type Reference Sets



Language type Reference Sets can be set up in a similar fashion to the above, with the following exceptions:

The *LanguageCode* field in the *Subset* record should be used to link the *Reference Set's concept* into the appropriate place in the *| Language type| metadata sub-hierarchy*. For example, a value of "en-US" in the *LanguageCode* field would result in the *Reference Set's concept* being created under *|US English|*:

SNOMED CT Model component

Foundation metadata *concept*

Reference Set

Language type

English

US English

RealmId

ContextId

- Where the *SubsetType* is "Language" and the *LanguageCode* is a single level (e.g. "en"), then the *Reference Set* should be created at the first level, under |English| in the example above;
- Where the *SubsetType* is "Language" and the *LanguageCode* is a two level (e.g. "en-US"), then the *Reference Set* should be created at the second level, under |US English| in the example above;
- Where the *SubsetType* is "Realm Description", then the *Reference Set* should be created under *RealmId* in the example above (where *RealmId* is the value of the *RealmId* field in the *Subset* record);
- Where the *SubsetType* is "Context Description", then the *Reference Set* should be created under *ContextId* in the example above (where *ContextId* is the value of the *ContextId* field in the *Subset* record and *RealmId* is the value of the *RealmId* field in the *Subset* record).

The *Reference Set* member records should be created as described in the following table:

Table 286: Converting a Language Subset to a Language Reference Set

Field	Data type	How to populate
<i>id</i>	<i>UUID</i>	A new unique <i>Identifier</i>
<i>effectiveTime</i>	<i>Time</i>	The nominal date on which this release was made.
<i>active</i>	<i>Boolean</i>	'1'
<i>moduleId</i>	<i>SCTID</i>	Set to the <i>moduleId</i> of the authoring organization .
<i>refsetId</i>	<i>SCTID</i>	A reference to the <i>concept</i> describing the <i>Reference Set</i> that you've just created.
<i>referencedComponentId</i>	<i>SCTID</i>	Set to <i>MemberId</i> in the <i>Subset Members</i> table record.

A Descriptor can also be set up if required.

9.2.1.4.8 Metadata hierarchy



As the *RF2* data structures and extensibility mechanism contain a number of *concept* enumerations, it is necessary to define *concepts* that represent these values. As well as the enumerated values, there are other machine-readable *concept model* structures not visible in the *Release Format* that require metadata (for example, the structures that define the format of a *description* type).

To meet this need, a new top-level *hierarchy* has been defined as a sibling to the | *SNOMED CT Concept* |, called | *SNOMED CT Model component* |. Note that existing metadata *concepts* held within the | *SNOMED CT Concept* | *sub-hierarchy* (|Linkage| and | Namespace |) will be moved to the | *SNOMED CT Model component* | *sub-hierarchy*.

The top level of the *SNOMED CT Model component hierarchy* is structured as follows:

- 138875005 | SNOMED CT Concept (SNOMED RT+CTV3) |
 - 900000000000441003 | SNOMED CT Model Component (metadata) |
 - 900000000000442005 | Core metadata concept (core metadata concept) | ...
 - (*Concept enumerations* required to support *SNOMED CT International Release* data structures)
 - 900000000000454005 | Foundation metadata concept (foundation metadata concept) | ...
 - (metadata required by the *Reference Set* extensibility mechanism)
 - 106237007 | Linkage concept (linkage concept) | ...
 - 246061005 | Attribute (attribute) | ...
 - 416698001 | Link assertion (link assertion) | ...
 - 370136006 | Namespace concept (namespace concept) | ...

Figure 126: SNOMED CT Model Component Hierarchy

Note that only | is a | *relationships* will exist between *concepts* in the | SNOMED CT Model Component || *hierarchy* |. Other associations between *concepts* in this *hierarchy* can be modeled using an | Association type reference set (foundation metadata concept) | (see [Association Reference Set](#)).

9.2.1.4.9 SCTIDs and UUIDs



UUIDs are unique universal *Identifiers*. These 128 bit unsigned *integers* can be used to identify all *SNOMED CT components* internally.

SCTIDs will continue to be used as primary and foreign keys for *concepts* and *descriptions*, both to identify a component and to reference other components. This form is essential for vendors and implementers who will reference *concepts* in *Clinical Information Systems* and messages. SCTIDs will also be used to identify *relationships*. However, UUIDs will be used to identify *Reference Set* members.

In addition, any UUIDs used in development can also be published as additional *Identifiers* via the *Identifier file*.

9.2.1.4.10 Description text



The values permitted within the *description term* field have been extended to support arbitrary length content, and support mark-up content such as XHTML. The *Description Type Reference Set* allows a maximum length and format to be associated with each *description* type within the *Description file*.

This mechanism allows descriptive text of different formats (other than *Fully Specified Names* and *Synonyms*) to be associated with *concepts*, while appropriately constraining existing *description* types. This enables all *descriptions* associated with *concepts* that may require translation to be held in one place in the *Description file*.

9.2.1.4.11 LanguageCode



The languageCode field is retained in the *Description file*, but is restricted to contain only coarse-grained *language* information (e.g. "English" or "French"). Reference sets are used to indicate *dialects* and contexts, where required. As an example, the *term* "Bulldozer" would appear once in the *Descriptions* file with the *language* code en ("English"), but be listed separately in each of the Australian, UK and US English *language* national *dialect Reference Sets* as a valid *term* in all three *dialects*.

The languageCode field in *RF2* is a text field and is bound to the ISO 639-1 two-character *language* codes.

9.2.1.4.12 Addition of a modifierId field



The underlying semantics on which *SNOMED CT* is based assumes that all *relationships* are existential restrictions. In other words, a *relationship* in *SNOMED CT* implies that there be **some** instance of that *relationship* from each instance of the source *concept* to any instance of the target *concept*. Other types of *relationship*, such as universal restrictions do exist and have been studied extensively. For example,

the existence of a universal *relationship* in *SNOMED CT* would require that ***all*** instances of that *relationship* from each instance of the source *concept* be to an instance of the target *concept*.

As an example, take the following hypothetical *relationship* |Has child| between two concepts |Woman| and |Girl|:

| Woman | | Has child | = | Girl |

In *SNOMED CT*, the *relationship* is implicitly an existential *relationship*, that we can make explicit in the above syntax by adding the modifier "some:", as follows:

| Woman | **some:** | Has child | = | Girl |

This means that every instance of | Woman | has at least one | Has child | *relationship* that has as its target an instance of | Girl|. In other words, in our hypothetical world, every woman would have at least one daughter, but may also have any number of sons.

If the existential *relationship* were changed to a universal *relationship*, as follows, then the meaning would be changed:

| Woman | **all:** | Has child | = | Girl |

This means that, for every instance of | Woman |, all its | Has child | *relationships* must have a target of | Girl | . In other words, in our hypothetical world, women could only have daughters or no *children* at all, and could not have sons. This has a very different meaning from the existential *relationship* currently implied within *SNOMED CT*.

A new *modifierId* field has been added to the *Relationship file* to allow future expansion. This *concept* enumeration field will initially be set to | Some | to keep compatibility with the existing semantics of *SNOMED CT*. Widening the range of this field to include other values (such as | All |) would in future increase the expressive power of *SNOMED CT*. However, this is likely to come at the cost of an increase in reasoning complexity, leading to potential issues for classification tooling. Therefore, before extending the range of this field beyond | Some |, a test of the impact on tooling will need to be performed, and the results reviewed and approved.

Notes:

1. The *modifierId* field has been included at this stage as the *RF2* format is likely to be stable for at least a five year period, without addition or deletion of fields. Within that period it is anticipated that other *modifierId* values will be added. Therefore, although not fully implemented at this stage, this field has been included in the initial *RF2* specification as it represents an integral part of the *Description Logic* used by *SNOMED CT*.
2. Any expansion of *SNOMED CT* to include *relationships* with a *modifierId* set to a value other than | Some | will be discussed with *Members* first and approved by the *Technical Committee*.

9.2.1.4.13 Addition of *moduleId* field



A *moduleId* field has been added to help identify content and dependencies in a release. This enables release centers to compose a unified release (in a single set of *release files*) from a number of different modules, yet still identify the origin of content down to a row level within each of the releases. For example, this may be used to differentiate *SNOMED CT International* content, Australian Medicines terminology and Pathology content within the Australian *National* release. Currently this is only possible if all modules are assigned unique sub-*namespaces*, and content consumers parse *Identifier namespaces* to differentiate modules.

Components may move from one module to another within a particular *namespace*. Without a *moduleId*, there would be a need to retire a component in one *namespace*, and add another (with a new *SCTID*) to the *namespace* that the component is moved to. Additional *relationships* would also need to be set up, to link the old and new components together. None of this administrative and error-prone work is required if *moduleIds* are used.

Combining the *moduleId* with *Reference Sets* provides a powerful versioning mechanism. The *Module Dependency reference set* (described in more detail in the *SNOMED CT Release Format 2 - Reference*

"SetSpecifications" document) can represent interdependencies between modules and define compatible versions. This functionality can thus be used to represent version information for a terminology's components within the terminology's content itself, in a machine processable way.

The diagram below provides an example of this structure. It shows the components making up an Australian national *SNOMED CT extension* release, containing subcomponents. The links can be described using members of the Module Dependency Reference Set. In the example below:

- *SNOMED CT Australian Extension* depends upon *SNOMED CT International* 2008-01-31;
- *Australian Pathology Extension* depends upon *SNOMED CT Australian Extension* 2008-08-31;
- *Australian Discharge Summary Extension* depends upon *SNOMED CT Australian Extension* 2008-08-31.

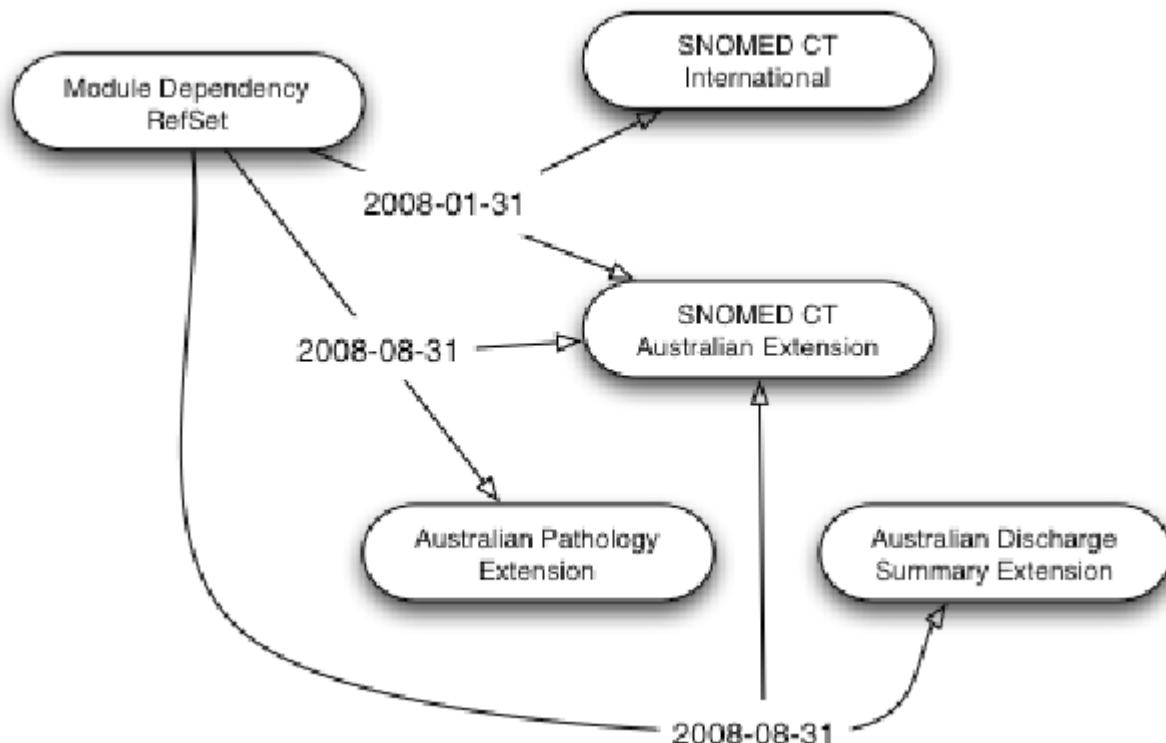


Figure 127: Illustration of module dependencies

9.2.1.4.14 Fully Specified Names and Preferred Terms



RF2, like the original *Release Format*, allows *Fully Specified Names* (FSNs) to be specified in each *language* using the *Description* file. Multiple FSNs and multiple *synonyms* may exist with the same *languageCode* for a *concept*. However, a particular *language Reference Set* will only contain a single FSN and a single *preferred term* for a *concept*.

As part of the *language* modifications made in the *RF2*, only a broad definition of a *language* can be made for a *Description*. For example, it is possible to declare a *Description* as English, but not US English. Also *RF2* no longer contains a *description* type value for a "*Preferred Term*", only types of | *Fully Specified Name* | and | *Synonym* |. Each *Synonym* can then be assigned an |*Acceptability* value| of either |*Acceptable*| or |*Preferred*| when included in a *language reference set*.

As a result of these changes, the preference for particular *descriptions* in a *language* or *dialect* is now represented using a *Reference Set*. This matches the specified use of *Language Subsets* in *RF1*, and deliberately removes the deprecated approach applied in some releases where preferences were derived directly from the released *Descriptions* file.

Language reference sets also introduce the notion of overriding or inactivating particular *Descriptions* that may be appropriate in one *dialect*, but not appropriate in another dependent *dialect* or context. This is achieved

by allowing *Descriptions* that are inherited from a parent *language reference set* to be overridden in a *child language reference set*.

9.2.1.4.15 Field removals



A number of fields that appeared in the previous *Release Format* do not appear in *RF2*. These fields are listed in the table below, with an explanation of why each field has been removed and to where it has been moved. Note that where a *reference set* replaces a field, this *reference set* will be provided with the *RF2* distribution.

Table 287: RF1 fields that are not use in RF2

File	Field	Rationale for change	Moved to
<i>Concept</i>	CTV3Id	To avoid cluttering the <i>concept</i> table.	Moved to the CTV3 simple map <i>Reference Set</i> .
<i>Concept</i>	SNOMEDID	To avoid cluttering the <i>concept</i> table.	Moved to the SNOMED RT IDsimple map <i>Reference Set</i> .
<i>Concept</i>	FullySpecifiedName	This field duplicates one of the <i>fully specified names</i> represented in the <i>Description file</i> . This duplication has led to misunderstanding of the use of <i>fully specified names</i> in multi-lingual distributions of <i>SNOMED CT</i> .	The original FSN, which may be required for translation purposes, can be identified as the FSN for the <i>concept</i> that has the earliest <i>effectiveTime</i> .
<i>Relationship</i>	Refinability	As this information is only useful in some environments, it has been moved out of the <i>Relationship file</i> to avoid cluttering it.	Moved to the <i>Relationship refinability reference set</i> .

9.2.1.4.16 Identifier file



The *Identifier file* has been added to provide a standardized means of attaching co-referent *Identifiers* from many different schemes to *SNOMED CT components*. This provides a means to:

- link *UUIDs* and *SCTIDs*, and;
- add external *Identifiers* such as *LO/NC codes*, where these are truly co-referent; and;
- track history and organizational responsibility by linking old *SCTIDs* to new ones, where components are transferred from one name space to another, in order to allow uninterrupted use of the old *SCTIDs*.

This provides a mechanism for generically binding *SNOMED CT components* to an arbitrary number of alternative *Identifiers*. It is a more scalable solution than appending columns as needed to the *Concept file*.

Note that the *Identifier file* is not intended as a mapping solution. This structure is only intended to support cases where the external *Identifier* means exactly the same thing as the *SNOMED CT component* to which it is attached. For example, it is not envisioned that *ICD-9*, *ICD-10* or *CTV3* codes would be entered into this file.

The *Identifier file* is intended to provide a mechanism to represent external codes for *SNOMED CT components* where the meaning is exactly the same. For example, in the Australian Medicines terminology (AMT), *concepts* are "generated" from data sourced from the Therapeutic Goods Administration (TGA) and the TGA has an ARTGID for every therapeutic item. This mechanism allows the ARTGIDs to be attached directly to the

corresponding AMT *concept* when generated. In this instance, the *Identifier file* assists meeting the use case without burdening the *descriptions* file or *concepts* file with this content.

9.2.1.4.17 References table



In the previous *Release Format*, the *References Table* contained References from *inactive components* to other equivalent or related *components* that were *current* in the *Release Version* in which that *component* was inactivated. Each Reference indicated the nature of the *relationship* between the *inactive* and persistent component.

In *RF2*, this information will be held in a number of Association *reference sets* (see the "History Tables" section in this guide and the "SNOMED CT Release Format 2 - Reference Set Specifications" document for more detail).

9.2.1.4.18 Textual Descriptions



In the previous *Release Format*, a separate *Textual Descriptions* file held long *descriptions* (of up to 512 bytes, in plain text format). In *RF2*, these textual *descriptions* are transferred to the *Description file*.

9.2.1.4.19 Mapping



9.2.1.4.19.1 Mapping Overview



No bespoke mapping file structures (for example, CrossMapSets tables) have been defined in *RF2*. Instead, the simple map *Reference Set* pattern and alternate map *Reference Set* pattern should be used, in conjunction with other *Reference Set* patterns, to define *Reference Sets* for mapping purposes. See the "SNOMED CT Release Format 2 - Reference Set Specifications" document for more details.

9.2.1.4.19.2 Representing RF1 Cross~Maps in RF2



RF1 Cross~Maps that have a type of either one-to-one or one-to-many can be represented in *RF2* as described below. The type of an *RF1 Cross~Map* can be identified from the *MapSetType* field in the CrossMapSets table. The following values in the *MapSetType* field are possible:

Table 288: RF2 Cross~Mapping Type Representations

Value	Meaning	Examples	Mapped to <i>RF2</i>
1	One-to-one	ICD-O	Can be mapped automatically, as described below
2	One-to-many	ICD-9-CM	Can be mapped automatically, as described below
3	Alternate on-to-one maps	None known of	Can be mapped automatically. Further definition will be given if necessary.
4	Alternate one-to-many	None known of	May need manual intervention to map.

For *Cross~Maps* that have a *MapSetType* of either '1' or '2', first, create a *concept* under the |Complex map| *sub-hierarchy* to describe the Complex map *Reference Set*, in the following location:

SNOMED CT Model component

Foundation metadata concept

Reference Set

Complex map

MapSetRealId

Where *MapSetRealId* is set to the contents of the *MapSetRealId* field in the *CrossMapsSets* record of the *Cross~Map* to be represented in *RF2* format. Where the *MapSetRealId* field is blank or null, then an intermediate *concept* should not be created, and the *Cross~Map Reference Set concept* should be created as a direct child of *|Complex map|*. The *concept* should be created as follows:

Table 289: RF2 Cross~Map Versioning

Field	Data type	Set to
<i>id</i>	<i>SCTID</i>	A unique id in your <i>namespace</i> .
<i>effectiveTime</i>	<i>Time</i>	The nominal date of release for your <i>Cross~Map reference set</i> . The year of the nominal release should tie up with the year in the <i>MapSetSchemeVersion</i> field in the <i>Cross~Maps Sets</i> record.
<i>active</i>	<i>Boolean</i>	'1'
<i>moduleId</i>	<i>SCTID</i>	The module <i>Identifier</i> for your authoring organization .

Once the *concept* is created, add two *Descriptions* for the *FSN* and a *Synonym*.

Table 290: RF2 Cross~Map Metadata

Field	Data type	Set to
<i>id</i>	<i>SCTID</i>	A unique id in your <i>namespace</i> .
<i>effectiveTime</i>	<i>Time</i>	The nominal date of release for your <i>reference set</i> .
<i>active</i>	<i>Boolean</i>	'1'
<i>moduleId</i>	<i>SCTID</i>	The module <i>Identifier</i> for your authoring organization .
<i>conceptId</i>	<i>SCTID</i>	The <i>Identifier</i> of the <i>concept</i> describing the <i>Reference Set</i> that you've just added.
<i>languageCode</i>	<i>String</i>	The <i>language</i> of the <i>Description</i> .
<i>typeId</i>	<i>SCTID</i>	Create two <i>descriptions</i> , with each of the following types: <i> FSN </i> , <i> Synonym </i> .

Field	Data type	Set to
<i>term</i>	<i>String</i>	Terms for the FSN and the <i>Synonym</i> . The <i>Synonym</i> should be set to the <i>MapSetName</i> in the Cross~Maps Sets record. The FSN should be set to: <i>MapSetSchemeName</i> + "(" + <i>MapSetSchemaId</i> + ")" + " reference set (foundation metadata concept)".

Finally, add members to the *Reference Set* that you've just created.

To do this, identify each CrossMaps table record with a *MapSetId* that matches the *MapSetId* field in the CrossMapsSets record for the *Cross~Map* that you're representing in RF2. For each CrossMaps table record, identify the related CrossMapTarget record using the *MapTargetId* field in the CrossMaps record. The *TargetCodes* field in the CrossMapTarget record will contain zero or more *target codes*, each separated by a separator character identified by the *MapSetSeparator* field of the CrossMapSets record.

One *Reference Set* member record should be created for each *target code* identified within the *TargetCodes* field, as follows:

Table 291: RF2 Cross~Map Representation

Field	Data type	Purpose
<i>id</i>	<i>UUID</i>	A unique <i>UUID</i> for the new member record.
<i>effectiveTime</i>	<i>Time</i>	The nominal date of release that this member is to be first introduced in.
<i>active</i>	<i>Boolean</i>	'1'
<i>moduleId</i>	<i>SCTID</i>	The module <i>Identifier</i> for your authoring organization .
<i>refsetId</i>	<i>SCTID</i>	The id of the <i>concept</i> that describes the <i>Reference Set</i> that you've just created.
<i>referencedComponentId</i>	<i>SCTID</i>	Set to the <i>MapConceptId</i> in the CrossMaps record.
<i>mapGroup</i>	<i>Integer</i>	This field should be set to '1' for the first <i>target code</i> within <i>TargetCodes</i> field of the CrossMapTargets record. If there is more than one <i>target code</i> in the field (separated by a separator character), then this field should be set to '2', '3', etc. For each subsequent code.
<i>mapPriority</i>	<i>Integer</i>	'1'
<i>mapRule</i>	<i>String</i>	Set to null
<i>mapAdvice</i>	<i>String</i>	Set to null

Field	Data type	Purpose
mapTarget	String	Set to the <i>target code</i> in the <i>TargetCodes</i> field of the <i>CrossMapTargets</i> record.

9.2.1.4.20 Release Types



Release Format 1 only supports a single *Release Type* which represented the entire set of currently relevant components. In contrast *Release Format 2* supports three different *Release Types* including a full historical view of all components ever released and a *delta release* that contains only the changes from one release to another.

The [Release Format 2 Specification](#) describes the *Release types* and the [Terminology Services Guide \(7\)](#) provides advice on *importing different Release types*.

9.2.1.4.21 Interchange format



RF2 is conceived as a replacement for the *current Release Format*. It is designed to provide a way to publish releases of *SNOMED CT Release* to implementers and other licensees. There is a close relationship between the requirements to support distribution of content and the requirements for exchanging components during content development. However, there are also significant differences related to the requirement for additional development information (author, change time, etc) and a need to support work with 'interim' incomplete and unpublished components which have not yet been assigned a *SNOMED CT identifier*.

Previous *IHTSDO* work resulted in a draft specification of *SNOMED Interchange Format (SIF)* which addressed some of these issues. Some of the provisions of *RF2* are already supported by SIF but others will require revisions to the SIF specification.

9.2.1.4.22 Post Coordinated expression Syntax



RF2 allows *relationship types* to be extended from "existential qualification" to other types of *relationship* such as "universal qualification". This *extension* will not be used in initial releases until the complexity of the underlying semantics has been fully tested, but once it is introduced, post coordinated expression syntax will also need to be extended to cater for this.

9.2.2 RF1 Compatibility and Conversion Tools



In January 2012 the *IHTSDO* switched from the original *Release Format* (used for *SNOMED CT* distribution since 2002), to the more flexible and consistent *Release Format 2* (*RF2*). This means that from that date onward the primary source data for the *SNOMED CT International Release* is maintained and distributed in the *RF2* format.

The *IHTSDO* recognizes that, while implementers will benefit from the features of the new format, there is inevitably a transitional period during which both formats are in use. Therefore, the *IHTSDO* provides the following resources to support users whose system do not yet support *SNOMED CT Release Format 2*:

- *Release Format 1* files will continue to be included in the *International Release* for a limited period
 - These files are not the authoritative version of *SNOMED CT* but are generated from the authoritative *RF2* data using a software utility developed for this purpose.
 - The resulting *RF1* data retains the functionality of the original release data but does not support any of the features of *RF2*. While all the clinically relevant *SNOMED CT* hierarchies are identical in both releases, the additional "Metadata Hierarchy" added as part of the *RF2* upgrade is not included in the *RF1* converted data. In addition there are some cases where *Identifiers* of *RF1* derivatives (Subsets and Cross-Maps) differ from those used for the equivalent *Reference Sets* in *RF2*. These differences are an essential consequence of ensuring that the *RF1* data produced by conversion from *RF2* is fully compatible with existing *RF1* systems.

- The RF2 to RF1 Conversion Tool used for generating the RF1 files is also available to all *IHTSDO Members and Affiliate Licensees*
 - The "RF2 Conversion Tool" is an open source, Java-based, software tool to facilitate the conversion of *SNOMED CT* files released in RF2 format into RF1 format. The tool provides both a command line utility and a Graphical *User Interface* (GUI) to facilitate configuration, progress tracking and the maintenance of additional data whenever it is not available as part of an RF2 release.
 - The limitations of RF2 to RF1 conversion (noted above) will also apply to conversion undertaken using this tool. To enable the conversion to be completed successfully in a way that retains and replaces *Identifiers* consistently for the RF1 environment a set of auxiliary files (the "RF1 Compatibility Package") is also required.

The "RF2 to RF1 Conversion Tool" and the "RF1 Compatibility Package" are available for *IHTSDO Members and Affiliates* to download in the same way as the *SNOMED CT International Release*.



Caution:

These resources and tools are intended for use during a transitional period and should not be considered as a long term alternative to migration to support direct use of RF2 data within applications. As *SNOMED CT* continues to evolve more of the specific feature of RF2 will be used to add value to the terminology. Some of the added value delivered by RF2 is soon likely to be regarded as essential for effective solutions to user requirements.

9.2.3 SNOMED CT identifier Update Notes



These notes provides update guidance on a change to the management and usage of *SNOMED CT Identifiers* agreed during 2011. The resulting changes to specifications and associated implementation guidance have been incorporated within the relevant sections of the Technical Implementation Guide from 2012-01-31.

The change described by this note is designed to remove the need for changing *SNOMED CT Identifiers* when transferring responsibility for maintenance and distribution of a *SNOMED CT component* from an *Extension* to the *International Release*, while maintaining an effective track of the origin and maintenance responsibilities for each *component*.

In addition, the change also remove the need for changing *SNOMED CT Identifiers* when transferring responsibility for maintenance and distribution of a *SNOMED CT component* from an *Extension* to an *Extension* that is a formally recognized hierarchical-ancestor of the originating *Extension*.

Rationale for the Change



SNOMED CT Identifiers are unique integer *Identifiers* which include embedded information about the type and origin of the *components* they identify. One part of this embedded information is the *namespace-identifier* which identifies the *Extension* in which the *component* originated.

Prior to the change described by this note the *namespace-identifier* also determined the organization responsible for maintaining the *component*. As a consequence, the specifications required that whenever responsibility for maintenance of *component* was transferred this required it to be inactivated and replaced by a new *component* with a new *SCTID* with a *partition identifier* and *namespace-Identifier* appropriated to the new maintenance arrangement.

The *Identifier* change resulting from moving a *component* from an *Extension* to the *International Release* causes disruption in the authoring environment. From an implementation perspective several *SNOMED CT Identifiers* changed from one release to the next, without any change in intended meaning, as a result of adoption of *concept* from an *Extension* as part of the *International Release*. These changes had a negative impact on system operation and interoperability between systems.



Description of the Change

The *namespace-identifier* continues to identify the *Extension* in which the *component* originated. However, it no longer implies a permanent immutable responsibility for maintenance. Instead, within specified limits and with agreement between the responsible organizations, the maintenance responsibility may be reassigned without issuing a new *Identifier*.

The permitted reassessments of responsibility are limited to ensure that the organization responsible for maintaining a *component* can be determined. Thus the end result of any transfer of responsibility must result in the *component* being maintained by one of the organization responsible for one of the following:

- the *Extension* namespace specified by the *namespace-identifier* of its *Identifier*;
- an *Extension* with a *namespace-identifier* that is a hierarchical ancestor of the *namespace-identifier* of the originating *Extension*;
- the *International Release*.

The values of the *partition-identifier* which previously indicated that a *component* was part of an *Extension*, continue to indicate that the *SCTID* contains a *namespace-identifier*. However, some *components* with a *namespace-identifier* may now be maintained as part of the *International Release*. Therefore, for clarity, the definition of *partition-identifier* has been revised to indicate that the values determine whether the *SCTID* conforms to the *long format* (with a *namespace-identifier*) or the *short format* (without a *namespace-identifier*). Only the IHTSDO can issue *short format* *SCTIDs* (without a *namespace*), whereas an *Extension* owner must issue a *long format* *SCTIDs* (including a *namespace-identifier* that is registered as belonging to them).

The *moduleId* field, introduced in *Release Format 2* and held against each *component*, records the organization currently responsible for maintaining the *component*. The *moduleId* must refer to a module delivered by the organization maintaining the *component* and the *namespace-identifier* of this *moduleId* must belong to the maintaining organization.

Following the change, migration of *components* between *Extensions* would be possible without a change to their *SCTIDs*, according to the following rules:

- A *component* can be moved from any *Extension* to the *International Release* without a change to its *SCTID*.
- A *component* can be moved from an *Extension* to a parent or ancestor *Extension* without a change to its *SCTID*.
- A *component* can be moved from the *International Release* back to its originating *Extension* without a change to its *SCTID*.

In all other cases, the existing rules for moving *components* between *Extensions* should be used. These require a change of *SCTID* to occur with tracking of inactivation in the appropriate *Component Inactivation Reference Set* and cross-references created in the appropriate *Historical Association Reference Set* (or as *historical relationships* in *Release Format 1*).

Caution: Components that originated in the *International Release* must **not** be moved to any *Extension* without a change to the *SCTID*.

In order to make explicit which *Extensions* are parents of which other *Extensions*, *concepts* under the *Namespace Concept* may now be rearranged as a nested hierarchy of namespaces. All namespaces at the top level of this hierarchy are considered to have as their parent the *International release*. The *Namespace Concept* for an *Extension* that is dependent on another *Extension* may be nested as a child (sub-type) of the *Namespace Concept* for the *Extension* on which it depends.

Caution: Components that originated in an *Extension* must **not** be moved to any other *Extension* unless the *Namespace Concept* associated with the target *Extension* is an ancestor of *Namespace Concept* associated with source *Extension*. The ancestry of *Namespace Concepts* is determined by the *subtype hierarchy* distributed as part of the *International Release*. Other moves between *Extensions* require a change of *SCTID*.

Guidance has been developed for producers and consumers of SCTIDs, to help avoid conflicts of ownership and to facilitate identification of owning organizations (see [Guidance for Producers of SNOMED CT Identifiers](#) and [Guidance for Consumers of SNOMED CT Identifiers](#)).

Benefits of the Change



The key benefits of making this change are:

- Large scale retirement and replacement of SCTIDs place an increased maintenance burden on implementers with no perceivable benefit. This change significantly reduces that burden.
- The change maintains the distinction of the namespace and module *Identifiers* - the former for the creators of content and the latter for the maintainers.
- The change eases the burdens of content providers in the chain of submissions to National *Extension* and the IHTSDO in detecting their content in public releases. It enables them to set policies on how to detect and manage content migration.
- Long term contributions will come from existing *Extensions*. This change will reduce impact on both the *National Release Centre Extension* managers and the source providers.
- The change removes the disincentive to migrate content to the *International release* or to a parent *Extension*.
- It will enable more frequent incremental release of content due to decreased migration burden.

9.3 Migrating Existing Data



The transition to *SNOMED CT* from legacy code systems²³ requires several changes. Many of the most important changes relate to organization and user training, which are outside the scope of this technical guide.

From the technical perspective, there are two principal *migration* issues.

- Maintenance of the integrity and value of pre-existing data recorded using other coding schemes (legacy data).
- Maintenance and development of the functionality delivered by software applications that use queries and protocols that include or refer to codes in other coding schemes (legacy queries and protocols).

9.3.1 Intended audience



The intended audience for this section is individuals or any organizations that wish to develop and deploy systems that use *SNOMED CT* but who currently have existing data represented using other coding schemes. This section is therefore contains information that is relevant to various people including:

- **Clinical software developers**, including those who have worked with a version of the Clinical *Terms* (the *Read Codes*) or with *SNOMED CT* terminologies;
- **Clinicians**, whose patient data has been stored in non - *SNOMED CT* systems and who rely on reports and decision support from these systems;
- **Healthcare planners, managers and information specialists** who rely on the secondary use of coded clinical information.

9.3.2 Migration requirements



Migration is required to enable information originally recorded prior to introduction of *SNOMED CT* to be retrieved and reused within a *SNOMED CT enabled application*.

Types of information that need to be considered as part of the migration process include:

- Coded data stored in existing systems.

²³ A "Legacy code system" is code system used prior to implementation of *SNOMED CT*.

- Information systems, e.g. software and hardware.
- Decision support protocols.
- Data entry templates.
- Queries and other data retrieval, aggregation and analysis specifications.

9.3.3 Strategies for migration



Moving from a legacy coding scheme to *SNOMED CT* requires attention to be paid to continued accessibility and use of data encoded using the legacy scheme.

The following general approaches may be applied or adapted taking accounts of the capability of the *SNOMED CT enabled application* and the value and relevance of existing data.

- Mapping or converting the data:
 - This requires each instance of a code in the existing data to be mapped to an appropriate *SNOMED CT expression*. This *expression* is then associated with the existing coded record entry as part of a record entry that conforms to the information model used in the *SNOMED CT enabled application*.
- Linking or integrating existing data:
 - The existing data is retained in its native form or in an intermediate form. However, the *SNOMED CT enabled application* is designed (or adapted) to access this existing information as if it had been converted. To deliver this functionality, queries and/or data are in effect mapped at the time of retrieval, rather than at the time of upgrading the system. This can be achieved in different ways some of which involve direct use of mapping tables and others in which, while the existing data is unchanged, an index derived from a map to *SNOMED CT* is generated to optimize subsequent access.
- Archive or retain old data in its original form, and where it is necessary to retrieve historical information, use *components* from the legacy system to do this:
 - This approach completely separates the new *SNOMED CT* from the legacy data and is unlikely to be acceptable in clinical practice. However, it may be appropriate for some data warehousing applications where the wholesale conversion of data is considered too onerous.

9.3.4 General considerations for data migration



A substantial body of clinical information resides in electronic systems, represented using existing coding schemes, terminologies and classifications. This information may be of value to individual patient records or to population aggregations. Similarly, there are many queries and decision support protocols that contain knowledge representation based on existing terminologies. The volume and heterogeneous nature the existing data means different approaches may be required to meet specific sets of requirements.

Factors that need to be considered include:

- The volume and value of existing in the context of the anticipated uses of a future *SNOMED CT application*:
 - The scale of the task and the potential value of migrated data are interrelated. Relatively small amounts of data that are of debatable value to a future system may not justify an elaborate migration process. On the other hand, it is vital to ensure that valuable existing data remains fully accessible within a *SNOMED CT enabled application*.

 **Example:** In the UK alone, there are over 50 million patients primary care electronic records coded using one of the versions of the *Read Codes*. Based on typical patterns of use this means there are several billion coded record entries that may need to be taken into account in the migration process. A substantial proportion of this data has continuing clinical value and thus despite the scale of the task it is important to ensure that data is migrated accurately and efficiently.

- Data quality and consistency:
 - Different users in different settings may select codes and terms in idiosyncratic ways to reflect their needs. This may be acceptable locally but it creates an obstacle to migration if the goal is consistent and comparable information at a regional, national or global level.
- Different source code systems:
 - Several different coding scheme versions are in use and each of these poses specific challenges and offers a different profile of potential benefits.
- Different information systems:
 - There are many system suppliers. As a result of system development and commercial mergers and takeovers, many suppliers support more than one application in the same domain. The challenge is to migrate from this diverse situation to a next generation environment supporting standards such as *SNOMED CT*.
- Different information models:
 - In addition to differences in the use of codes, existing systems inevitably have a variety of approaches to structuring clinical information. As a result, the process of migrating data between systems is not simply a question of converting codes. The underlying architecture of the source data also needs to be taken into account to make optimal use of existing data without losing processable information or introducing errors.

9.3.5 Specific data migration issues



9.3.5.1 Retaining existing coded data



Migration does not mean over-writing legacy coded data with *SNOMED CT* expression. **This is strongly discouraged** and users are advised to ensure that data stored at the time of data entry is preserved. This is essential for two main reasons:

- Medico-legal status of an altered clinical record may become degraded;
- The original record may be an invaluable resource, should *migration* produce unexpected results.

9.3.5.2 Hierarchies and Identifiers



The use and representation of hierarchies in *SNOMED CT* differs from the approach taken in many older code systems and classifications. This has a number of consequences that may affect the migration of queries, decision support protocols and data entry templates.

- Meaningless *Identifiers*:
 - The codes specified in *ICD-9*, *ICD-10*, the *Read Codes* and *SNOMED International* provide information about where the code is located in a hierarchy. This allowed simple pattern matching queries to be used for some types of retrieval.

👉 **Example:** In *SNOMED International*, all 'diagnoses related to the digestive system' can be retrieved by a query for all codes starting with 'D5'

The *Identifier* of a *SNOMED CT concept* does not provide any information about the way it relates to other *concepts*. Therefore, a simple pattern matching query cannot be used to retrieve related information represented using *SNOMED CT*. Instead, the query that specifies a *subtype* of the required *concept* is evaluated by testing the *transitive closure* of the set of *subtype Relationships*.

👉 **Example:** All 'diagnoses related to the digestive system' can be retrieved by a query for *expressions* that are *subtypes* of 119292006 | disorder of gastrointestinal tract |.

- Polyhierarchy:
 - *Statistical classifications* and many other code systems have a *monohierarchy* in which each code falls within only one branch of the hierarchy. In contrast, the *SNOMED CT subtype hierarchy* is a *polyhierarchy*, which means that each *concept*s can be a *subtype* of many different *concept*s. This is a powerful feature of *SNOMED CT* but it may significantly alter the results of a migrated from an earlier scheme.
 - 👉 **Example:** All 'diagnoses related to the digestive system' in some schemes may exclude codes that are primarily classified as infective disorders even they affect the digestive system. However, a *SNOMED CT* query includes all *subtype concepts* regardless of whether they are also in another hierarchy.
- Different hierarchies:
 - Hierarchies in different code systems may be based on different principles and as a result queries that are migrated to *SNOMED CT* may return unexpected results. This may also relate to different interpretation of apparently identical *concept*s.
 - 👉 **Example:** Should a query for the *concept* 'nephrectomy' return only patients who have had a total nephrectomy or should the results include those with a record of a partial nephrectomy? If the partial nephrectomies are to be included then how much kidney tissue must be removed to count as a nephrectomy? If the partial nephrectomies are not included then should a record entry including the *concept*'nephrectomy' (without specifying partial or total) be included? The *SNOMED CT subtype hierarchy* determines the answer to these questions based on the defining *Relationships*. The query may need to be refined to meet more specific expectations.

9.3.5.3 postcoordination



SNOMED CT, enables the use of *postcoordinated expressions* to represent detailed clinical information (such as observations or procedures) by reference to multiple *Concept Identifiers*.

When considering migration of existing data an important question is whether *postcoordination* is required to replace existing coded data. The answer to this question depends on the specificity and expressivity of the existing coding scheme.

There are four situations in which *postcoordination* may be required or useful in the mapping process.

- To capture data *postcoordinated* in the original coding scheme:
 - The extent to which this data can be mapped to *SNOMED CT* depends on the consistency of the original representation and the degree of alignment with the *SNOMED CT Concept Model*. Refinements in the source data are not sanctioned by the *Concept Model* may be mapped to similar approved *Attributes* or downgraded to text. Alternatively, they may be retained as *expressions* that fall outside the scope of *Concept Model* although this may limit effective retrieval.
 - 👉 **Example:** Information coded using *NHS Clinical Terms Version 3*, may also include *postcoordination* using *qualifiers*. In most cases, these *qualifiers* can be represented using *postcoordinated expressions*.
- To represent information which the originating coding system precoordinates but which *SNOMED CT* can only represent using a *postcoordinated expression*.
 - 👉 **Example:** A specialized radiology coding systems may have separate codes for the same procedure applied to different body sites. If *SNOMED CT* does not include these, they can be represented using *postcoordinated expressions* with 'procedure site' and 'laterality' *Attributes* applied.
- To incorporate additional information which the originating clinical system represents in a consistent proprietary form.

👉 **Example:** A system may have a separate field for laterality and this can be applied during the mapping process to generate a *postcoordinated expression*.

- To satisfy a preference for a consistent *postcoordinated* representation of a particular type of data.

👉 **Example:** There may be a preference to always represent allergies by postcoordinating the substance, rather than using one of the *precoordinated* 'allergic to x' concepts.

9.3.6 Migration from earlier SNOMED CT code systems



This section contains specific advice related to migration to *SNOMED CT* from previous *SNOMED CT* code systems including *SNOMED RT* and *SNOMED International*.

9.3.6.1 Migration from SNOMED RT



Migration from SNOMED RT poses very few significant issues, since many features of the design of *SNOMED CT* including the use of *SCTIDs* were incorporated into *SNOMED RT*.

The transition to *SNOMED CT* for users of *SNOMED RT* is relatively straightforward because the *Concept Identifiers* of *SNOMED RT* are for the most part the same as those used in *SNOMED CT*. In some cases, during the merger of *SNOMED RT* and *Clinical Terms Version 3* some *Concepts* in *SNOMED RT* have been found to be ambiguous or duplicated. These *Concepts* have been inactivated by an appropriate change of status and adding a record in the Component Inactivation Reference Sets, but are still present in the *Concepts Table* and are linked to *Active Concepts* by the Historical Associations Reference Set.

- Each duplicate *Concept* has a | SAME AS | Association to the *Active Concept* with the same meaning as the duplicate *Concept*;
- Each ambiguous *Concept* has | MAY BE A | Association to one or more *Active Concepts*, which represent possible disambiguated meanings.

These Associations can be used either to allow *Concepts* recorded using these *Concepts* to be recognized by retrieval tools or to enable mapping of the stored information to the appropriate *active Concept Identifier*.

If any stored *Concept Identifier* of an *Inactive Concept* is mapped to an *active Concept Identifier* using these Associations it is strongly recommended that the original *Concept Identifier* is also retained. This enables future improvements or corrections of such mappings if revised Associations are present in a future release of *SNOMED CT*.

In addition, *SNOMED RT* contained both generic and brand name drugs for the US. A decision was made during the merger process to not retire these *concepts* using the extension mechanisms, but to place these components directly in the US Drug Extension. Therefore to access all *SNOMED RT components* you will need to use the US Drug Extension in addition to the *SNOMED CT International Release*.

Like *SNOMED CT*, *SNOMED RT* also contains the appropriate legacy *SNOMEDID* from *SNOMED International*.

9.3.6.2 Migration from SNOMED International



The meaning of coded clinical data encoded using *SNOMED International* is maintained using mechanisms that support *concept permanence* and version control.

Concept permanence ensures that codes assigned in *SNOMED International* are retained, accessible, and not reused. The codes used in *SNOMED International* are present in *SNOMED CT* in the *SNOMEDID* field of the *Concepts Table*. Even when a *Concept* is *retired* from *active* use its code is never reassigned to another *Concept*.

The *SNOMEDID* in the *Concepts Table* can be used either to allow recognition of legacy data by *SNOMED CT* retrieval tools or to enable mapping of the codes and storage of the appropriate *Concept Identifier*.

9.3.6.3 Migration from earlier versions of SNOMED



To assist in the *migration* of legacy data from early version of SNOMED - SNOMED II (1979) and SNOP (1965), a Bridge File (or mapping table) is available from the IHTSDO which links the legacy code to its corresponding code in *SNOMED International*.

As noted in the previous section, *SNOMED CT* retains the codes used in *SNOMED International* in the *SNOMEDID* field of the *Concepts Table*. This can be used to complete the linkage or mapping of legacy data encoded using earlier versions of *SNOMED*.

9.3.7 Migration from Read Codes and CTV3



Organizations planning to migrate from NHS *Clinical Terms Version 3* or earlier versions of the *Read Codes* are advised to review the documentation and mapping tables published by the *UK NHS*.

Separate advisory documents and table for each of the *Read Code* versions are available as part of the NHS Terminology Reference Data Update Distribution Service (TRUD)

<https://www.uktcregistration.nss.cfh.nhs.uk/trud/>. These materials are updated with each *SNOMED CT UK Extension Release*.

Chapter 10

10 Extension Services Guide



This part of the guide describes additional services which some advanced users or implementers may require to allow them to create or maintain *Extensions* for use in a particular country, organization or specialty.

The most common of these requirements will be to support the creation and maintenance of specialized *Reference sets*. Uses for *Reference Sets* include representation of *value sets*, marking *descriptions* to indicate acceptability of *terms* in a specific *language* or specialty, alternative hierarchies, *cross mapping* to classifications and *annotations*.

10.1 Rationale for Extensions



An *Extension* mechanism allows authorized organizations to add locally valid content and *Reference Sets* without compromising the main body of *SNOMED CT*. This facility will be valuable to:

- Meet the needs of specialties and *realms*;
- Meet vendor needs;
- Meet local business needs.

10.2 Extension Namespaces and SNOMED CT identifiers



The *components* of an *Extension* have *Identifiers* (*SCTIDs*) which have the same structure as those used in the *SNOMED CT International Release*. However, these *Identifiers* include a *partition-identifier* indicating that the *component* is part of an *Extension* and a *namespace identifier* specific to the responsible organization .

Partition-identifiers and *namespace identifiers* serve two roles:

- Prevention of *Identifier* collision or reuse:
 - Organizations responsible for an *Extension* must only issue *components* within their allocated *namespace* and must not reuse any *Identifier* within that *namespace* once it has been issued.
- Indicating the source for information about an identified *component*.
 - If an application receives instance data containing an *Identifier* that it does recognize , the application can use the *namespace identifier* to determine the responsible issuing organization .
 - The responsibility for an allocated *namespace* remains with the organization to which it was issued unless responsibility is transferred by merger or mutual agreement. Any *namespace* transfer must be notified to and authorized by the *IHTSDO*.

Guidance for Producers of SNOMED CT Extensions



Prerequisites

Before generating *SCTIDs*, an organization must own a namespace. A namespace can be requested from IHTSDO by emailing a request to info@ihtsdo.org.



Guidance on Generating SCTIDs



The following guidance is provided for owners of namespaces that generate new content:

- An organization should only generate new *SCTIDs* for *components* within a namespace that they own.
- An organization should have a mechanism in place to ensure that *SCTIDs* are not assigned multiple times. Generally, a single authority that generates *item-identifiers* in a sequential fashion for each type of *component* will achieve this goal.
- Generally, *SCTIDs* should be generated for new *components* as part of the release process for an *Extension*, rather than during the edit process. This is to avoid unnecessary usage of *Identifiers* for *Concepts* that are created during editing but found not to be required prior to release.
- *item-identifiers* should not be generated so as to have meaning. They should be regarded as meaningless numbers.

Guidance on Packaging Content



Organizations may package content into *release files* in a number of ways:

- All content for a particular type of *component* (e.g.: of type *Concept*) that is owned by the organization can be released in a single file. *Components* in this file may have different *moduleIds*, where the content has been authored by more than one group in the organization and each group has its own *moduleId*. Content that is owned by parent organizations may be held in separate files that are also included in the release. Content owned by other organizations should not be included in the release.
- As above, but *components* with different *moduleIds* can be released in separate files.
- As the first bullet above, but content from parent organizations may be included in the same *release files* as content owned by the releasing organization. In this case, the ownership of each *component* can be identified by reference to its *moduleId*. Care should be taken not to modify, add to or remove content that is owned by a parent organization, as this would be considered as editing content that the organization did not own.

Guidance on Promoting Components



Components (whether *Concepts*, *Descriptions* or *Relationships*) may be promoted to a parent *Extension* or to the *International release*. In order to achieve this, the donating organization should contact IHTSDO or the owner of the receiving *Extension* with details of the *components* that are to be promoted.

The definition of the *component* in the source *Extension* should not change (for example, a new record should not be added to the source *Extension* to deactivate the *component*). Once the *component* has been promoted to a parent *Extension*, care should be taken not to amend or deactivate it within its original *Extension*.

A *Component* should only be promoted to the *International release* or to an *Extension* that is associated with a namespace that is a parent of the namespace of the *component*.

Guidance on Receiving Promoted Components



Before receiving content into a parent *Extension* or the *International release*, details of the *components* that are to be transferred should be received in writing from an authority within the source organization.

The *component* should then be included in the next release of the parent *Extension* or of the *International release*, with the following fields amended:

- *effectiveTime* – to be set to the *Extension's* release date, as normal.
- *moduleId* – set to the *moduleId* of the new maintaining organization.

The *SCTID* of the *component* should not change when it is included in the new *Extension* or the *International release*.

Guidance on Defaulting Components to their Original Extension



Where a *component* has been promoted to a parent *Extension* or to the *International release* (perhaps incorrectly), and it is required to default that *component* back to its original *Extension*, then the parent organization should contact the owner of the *components' original Extension* with details of the *components* that are to be moved.

The *component* should then be included in the next release of the parent *Extension* or of the *International release*, with the following fields amended:

- *effectiveTime* – to be set to the *Extension's* release date, as normal.
- *active* – set to false.

At this point, the *component* will be retired for all consumers of the parent *Extension* or the *International release*.

The *component* should then be included in the next release of the original *Extension* to which the *component* is to be moved, with the following fields amended:

- *effectiveTime* – to be set to the *Extension's* release date, as normal.
- *moduleId* – set to the *moduleId* of the new maintaining organization.

The *SCTID* of the *component* should not change when it is included in the receiving *Extension*. Care should be taken to ensure that the *effectiveTime* of the inactivation record in the donating *Extension* is prior to the *effectiveTime* of the record in the receiving *Extension*. The donating *Extension* should always deactivate the *component* before it is included in the receiving *Extension*. In particular, the *effectiveTimes* should not be set to the same date in order to avoid a primary key conflict for the *component* across the *Extensions*.

The following example shows how a *Concept* can be created in an *Extension*, promoted to the *International release* and then be defaulted to its original *Extension* without changing its *SCTID*. In this example, *|Module 1|* is owned by namespace 0989121 and *|Module 2|* is owned by *IHTSDO*.

A *concept* is first created in *Extension* 0989121:

Extension for namespace 0989121:

SCTID	effectiveTime	Active	moduleId	definitionStatusId
1290989121103	20071031	1	Module 1	Primitive

It is then included in the *International release*. At this stage, *IHTSDO* owns the *concept*. Note that there is no need to deactivate the *concept* in the *Extension* as the *Extension* is dependent on the *International release*, and therefore can only be used in conjunction with the *International release*. Because of the state valid representation of RF2, the new *concept* version added to the *International release* automatically supersedes the previous *concept* version in the *Extension*. Also, *|Module 2|* supersedes *|Module 1|* as the new module in which this *concept* is now authored:

Extension for namespace 0989121:

SCTID	effectiveTime	Active	moduleId	definitionStatusId
1290989121103	20071031	1	Module 1	Primitive

International release:

SCTID	effectiveTime	Active	moduleId	definitionStatusId
1290989121103	20080131	1	Module 2	Primitive

Then, IHTSDO inactivates the concept within the *International release*:

Extension for namespace 0989121:

SCTID	effectiveTime	Active	moduleId	definitionStatusId
1290989121103	20071031	1	Module 1	Primitive

International release:

SCTID	effectiveTime	Active	moduleId	definitionStatusId
1290989121103	20080131	1	Module 2	Primitive
1290989121103	20080731	0	Module 2	Primitive

Finally, the original *Extension* owner can include the concept within their own *Extension* again. At this stage, the concept will be inactive to all consumers of the *International release* that do not also consume *Extension* 0989121:

Extension for namespace 0989121:

SCTID	effectiveTime	Active	moduleId	definitionStatusId
1290989121103	20071031	1	Module 1	Primitive
1290989121103	20081031	1	Module 1	Primitive

International release:

SCTID	effectiveTime	Active	moduleId	definitionStatusId
1290989121103	20080131	1	Module 2	Primitive
1290989121103	20080731	0	Module 2	Primitive

Note that in the above example, the concept must be explicitly inactivated in the *International release* and it is not adequate in this case simply to rely on the new concept version that was added in the *Extension* to supersede the old concept version in the *International release*. This is because the *International release* is not dependent on the *Extension*, and so consumers that are taking the *International release* without taking the *Extension* need to be aware that, for them, the concept has been inactivated.

Guidance on other Movement of Components between Extensions

All other movement of components between *Extensions*, including moving content that was created in the *International release* to an *Extension*, or from one *Extension* to another unrelated *Extension*

should be performed by inactivating the *component* from the source *Extension* and creating a new *component* in the receiving *Extension*, as is described in the *SNOMED CT Technical Reference Guide*.

The reasons for constraining movement of *components* between *Extensions* in this way are:

- To ensure that two *components* with the same *SNOMED CT identifier* are not released independently (and perhaps inconsistently) in two separate *Extensions*.
- To allow a consumer to validate that the owner of an *Extension* has the authority to release all the *components* included in their release.

If it were allowed for a *component* to be retired from the *International release* and moved to an *Extension* while keeping the same *SNOMED CT identifier*, then it would be possible for more than one *Extension* owner to include the *component* in their *Extension* (perhaps over a period of time). This would result in issue (a) above. IHTSDO would have no way of monitoring this, or providing guidance to consumers of *Extensions*, to allow them to validate ownership of *components* within an *Extension* (issue b above). More seriously, once a *component* is in two separate *Extensions* with the same *SNOMED CT identifier*, then it may get modified in different ways in each *Extension* over time, causing interoperability issues.

Guidance for Consumers of SNOMED CT Identifiers



Guidance on validating SCTIDs within an Extension



The following checks may be performed to validate the consistency of *SCTIDs* in one or more *Extensions*:

- An *Extension* should only contain *components* that have a namespace owned by the releasing organization, or a child of a namespace owned by the releasing organization. Note however, that a releasing organization may merge content from its *Extension(s)* with one or more parent *Extensions* and the *International release* into a single *release file*.
- The primary key for *component* versions held as rows in *release files* is the composite of the *SCTID* and the *effectiveTime*. No two *component* versions should have the same primary key, either within or across all *Extensions*. Once loaded, the state valid history of a *component* across all loaded *Extensions* should be taken in the normal *effectiveTime* order.
- If a child *Extension* releases a new version of a *component* that has not been inactivated within the parent *Extension*, then there is an error. The version of the *component* in the parent *Extension* should be taken as the correct version of the *component* (as they have not formally released control of it), and the error should be reported to the owner of the child *Extension*.
- The *check digit* of each *SCTID* may be validated using the *check-digit* algorithm.

The following provides examples of possible errors that can be picked up as part of a validation process:

Here, the *release file* contains a *concept* that has a namespace that is not a child or parent namespace of namespace 0009999.

Extension for namespace 0009999:

<i>SCTID</i>	<i>effectiveTime</i>	<i>Active</i>	<i>moduleId</i>	<i>definitionStatusId</i>
1290989121103	20071031	1	Module 1	Primitive

Here, the *concept* has been incorrectly inactivated in an *Extension* at the same *effectiveTime* as the new *concept* version has been included in the *International release*. A clash in primary keys of the two *concept* versions has resulted.

Extension for namespace 0989121:

<i>SCTID</i>	<i>effectiveTime</i>	<i>Active</i>	<i>moduleId</i>	<i>definitionStatusId</i>

1290989121103	20071031	1	Module 1	Primitive
1290989121103	20080131	0	Module 1	Primitive

International release:

SCTID	effectiveTime	Active	moduleId	definitionStatusId
1290989121103	20080131	1	Module 2	Primitive

Here, the *concept* has not been deactivated in the *International release* before it was reinstated in an *Extension*. This is an error that would result in consumers of the *International release* receiving the 31st January version of the *concept*, with consumers of *Extension 0989121* (and the *International release*) receiving 31st October version of the *concept*, resulting in risks to semantic interoperability. In this case, the 31st January version included in the *International release* should be taken as the correct version and the error should be reported to the owner of *Extension 0989121*.

Extension for namespace 0989121:

SCTID	effectiveTime	Active	moduleId	definitionStatusId
1290989121103	20071031	1	Module 1	Primitive
1290989121103	20081031	1	Module 1	Primitive

International release:

SCTID	effectiveTime	Active	moduleId	definitionStatusId
1290989121103	20080131	1	Module 2	Primitive

Guidance on identifying the maintaining authority for a component

Where information about a *component* is available (either from *release files* or from a *terminology server*), then the *moduleId* of the *component* can be used to identify its maintaining authority. As an example, take the following case, where a *concept* was created by the owner of namespace 0989121, but then subsequently transferred to IHTSDO:

Extension for namespace 0989121:

SCTID	effectiveTime	Active	moduleId	definitionStatusId
1290989121103	20071031	1	Module 1	Primitive

International release:

SCTID	effectiveTime	Active	moduleId	definitionStatusId
1290989121103	20080131	1	Module 2	Primitive

If a consumer wishes to know who is currently responsible for maintaining the *concept* with *SCTID* 1290989121103, then the *release files* may be used to establish ownership. As can be seen, the *moduleId* of the most recent version of the *concept* is |Module 2|, which would be identifiable as belonging to *IHTSDO* as it is a short format *SCTID* (which does not have a namespace). Similar information should also be available via *terminology servers*, where such services are available.

If only the *SCTID* is available, then the namespace embedded in the *SCTID* can be used to check ownership of a *component*. In that case, the maintaining authority is most likely to be the organization that owns the namespace of the *component*. If it is not, then the organization that owns the next namespace up in the namespace hierarchy should be checked, where *IHTSDO* is positioned as the ultimate parent in the namespace hierarchy.

Guidance on parsing and identifying SCTID s



The constraints on the value range for *SCTIDs* allow a consistent *string* and integer representation of these values. The upper limit of 18 digits ensures that any valid *SCTID* can be stored in either a signed or unsigned 64-bit integer. The lower limit of six digits ensures that a *SCTID* can be distinguished from:

- A *Read Code*, which is 5 characters in length, padded out with dots if necessary.
- A *SNOMED ID*, which always starts with a letter.

Guidance on using state valid data



When receiving data from an *Extension* owner, care should be taken when reviewing historical data only to use snapshots of data relating to one of the release points for that *Extension*. It is only at these release points that the content in the *Extension* is consistent with the content in the *International release* and/or any parent *Extensions*.

For example, take the case where the *International edition* is released in January and an *Extension* is released in April. Generally, the *Extension* will be dependent on the *International release*, and may, for example, hold a *concept* that is a child of a parent *concept* in the *International release*. Now, if the parent *concept* is amended or even retired in January, the child *concept* will need to be reviewed, modified and perhaps moved to another part of the hierarchy to take account of the changes in the *International release*. Generally, the reason that the *Extension* is released a few months after the *International release* and not earlier is that the *Extension* owner needs to review the changes in the *International release*, modifying the content in the *Extension* to keep it consistent. Once the April release is made, the *Extension* and the *International release* will be consistent, and before the April release, consumers of the *Extension* should use the previous (July) version of the *International release* supplied by the *Extension* owner. So once the April release is made, the *Extension* and the *International release* are consistent, but any historic state between January and April will be inconsistent. In practice, this is not a big issue, as no changes will have been made between January and April.

Guidance where RF1 format is used for an Extension



Where a namespace owner is still releasing an *Extension* using *Release Format 1* (RF1), then content should continue to be moved from and to that *Extension* by creating new *SNOMED CT identifiers* and using the old “move to / move from” mechanism.

Consumers of *RF1* releases or conversions of *Extensions* should be aware that content may have been moved to the *International Release*, or to a parent *Extension*, without a change of *SCTID*. Therefore, these *Extensions* and so may contain *components* from with different.

10.3 Editing and Maintaining Extensions



A future document will provide advice for organizations that are authorized to develop *Extensions*.

This section includes:

- General principals of the process of creating, maintaining and distributing *extensions*;
- Links to *IHTSDO Workbench* documentation described the practical processes involved.

Chapter

11

11 SNOMED CT file and field names



This section lists the file and field names used in technical specifications within this guide. The scope of use of these names is limited to the tables in which they are used and the given definitions are not intended for use in any other context.

A



acceptabilityId (field)



A field in a 900000000000506000 | Language type reference set | that indicates the acceptability of a *Description* in the language or *dialect* specified by that *Reference Set*. Values include "preferred" and "acceptable".

👉 Note: Field name in a 900000000000506000 | Language type reference set |

active (field)



A Boolean field specifies whether a *component* is an *Active Component* from the point in time specified by the *effectiveTime*.

👉 Note: Field name in SNOMED CT Release Format 2.

alternateIdentifier (field)



A field in the *Identifier file* containing the representation of an *Identifier* in another code system that is irrevocably linked to a *SNOMED CT identifier*.

annotation (field)



An Annotation Reference Set field containing additional information linked to a *SNOMED CT component*.

👉 Note: Field name in SNOMED CT Release Format 2.

attributeDescription (field)



A reference to a *concept* that specifies the name and/or usage of an additional attribute in a *Refset*. If the *attributeType* is component reference, the values applied to this additional attribute are restricted to *subtypes* of this *concept*.

👉 Note: Field name in a SNOMED CT Release Format 2 Reference Set Descriptor.

attributeOrder (field)



An integer representing the position of an additional attribute in a *Refset*. The value 0 (zero) refers to the *referencedComponentId*. All other values refer to the position of an additional attribute relative to the *referencedComponentId*.

 **Note:** Field name in a SNOMED CT Release Format 2 Reference Set Descriptor.

attributeType (field)



A reference to a *concept* that specifies the data type of an additional attribute in a *Refset*.

 **Note:** Field name in a SNOMED CT Release Format 2 Reference Set Descriptor.

B



Boolean (data type)



A datatype that represents either true or false.

 **Note:** In *SNOMED CT release files* the value 0 (zero) represents "false" and the value 1 (one) represents true.

C



caseSignificanceId (field)



A field in the *Description Release File* containing a *SNOMED CT identifier* that indicates whether the text of the term can be modified to by switching characters from upper to lower case (or vice-versa).

 **Note:** Field name in SNOMED CT Release Format 2

characteristicTypeId (field)



A reference to a *concept* that specifies the nature of a *Relationship*. Values include "defining", "qualifying" etc.

 **Note:** Field name in the SNOMED CT Release Format 2 relationships table.

Concept file



The file structure used to distribute *SNOMED CT concepts*.

 **Note:** Component File name in SNOMED CT Release Format 2

conceptId (field)



A field in the *Description file* that associates a *term* with the *concept* to which it applies .

 **Note:** Field name in the *Description file*.

correlationId (field)



A field in the Complex Map Reference Set containing a *SNOMED CT identifier* which represents the correlation between the *SNOMED CT concept* and the *target code*.

 **Note:** Field name in SNOMED CT Release Format 2

D



definitionStatusId (field)



A field in the *Concept Release File* containing a *SNOMED CT identifier* which specifies whether the *concept* is *fully defined* or *primitive*.

 **Note:** Field name in the SNOMED CT Release Format 2 concepts table.

Description file



The file structure used to distribute *SNOMED CT descriptions*.

 **Note:** Component File name in SNOMED CT Release Format 2

descriptionFormat (field)



A *Description Format Reference Set* field reference to a *concept* that specifies the format of the *term* fields for a particular type of *Description*.

 **Note:** By default the *term* is a *UTF-8* string of up to 255 characters. However, description types can be specified which are longer in length and/or contain format markup (e.g. HTML).

descriptionLength (field)



A *Description Format Reference Set* field containing an integer which indicates the maximum length of the term string for a specified type of *Description*.

 **Note:** Field name in SNOMED CT Release Format 2.

destinationId (field)



A field in the *Relationship Release File* containing a *SNOMED CT identifier* that refers to the *concept* that represents the destination (or *attribute-value*) of the associated *Relationship*.

 **Note:** Field name in SNOMED CT Release Format 2. In RF1 this field was called *ConceptId2*

Dualkey (field)



A key used to facilitate textual searches of *SNOMED CT* that consists of the first three letters of a pair of words in a *Description*. All possible pairs of words in each *Description* may be paired irrespective of their relative position in the *Description*. *Dualkeys* are represented as a row in the *Dualkeys Table*.

 **Note:** Field name in SNOMED CT toolkit

Dualkey table



A table in which each row represents a *Dualkey*. See [see *Word Search Tables - Summary* on page 126].

 **Note:** File or Table name in SNOMED CT toolkit

DescriptionTypeId



A field in the *description file* that specifies whether the associated term is a *Fully Specified Name*, a *synonym* or a *text definition*.

 **Note:** The *descriptionTypeId* does not specify whether a particular term is acceptable or preferred for use in a given language or *dialect*. This information is conveyed by the Language Reference Set for the relevant language or *dialect*.

 **Note:** Field name in SNOMED CT Release Format 2.

E



effectiveTime (field)



Specifies the inclusive date at which the component version's state became the then current valid state of the component.

 **Note:** Field name in SNOMED CT Release Format 2

Excluded word (field)



A word that in a given *language* is so frequently used, or has so poor a discriminating power, that it is suggested for exclusion from the indices used to support textual searches of *SNOMED CT*. *Excluded Words* are represented as a row in the *Excluded Words Table*

 **Note:** Field name in SNOMED CT toolkit

Excluded words table



A data table in which each row represents an *Excluded Word*. See [see [Word Search Tables - Summary](#) on page 126].

 **Note:** File or Table name in SNOMED CT toolkit

I



Identifier file



The file structure used to distribute alternative *Identifiers* for *SNOMED CT components*.

 **Note:** Component File name in SNOMED CT Release Format 2

id (field)



A field that provides the unique identifier of a *component* (*concept*, *description* or *relationship*) or *reference set member*.

 **Note:** The data type of the *id* for a *component* is *SCTID* but the data type of the *id* for a *reference set member* is *UUID*.

identifierSchemaId (field)

A field in the RF2 *Identifier* file containing a *SNOMED CT identifier* which identifies the alternate code system.

Integer (data type)

A datatype that represents a whole number.

Note: In *SNOMED CT release file* specifications integers are represented as a string of decimal digits. The range of values and support for negative values may be constrained for the specification are specified for each usage of this datatype. However, unless otherwise specified, all *release file* fields of data type *integer* are assumed to be 32-bit signed integers.

K**Keyword (field)**

A field containing a potential search text in one of the *WordKey Tables* or a word excluded for key generation in the *Excluded Words Table*.

Note: Field name in SNOMED CT toolkit

L**linkedTold (field)**

An Ordered Reference Set field containing a *SNOMED CT identifier* which refers to either a sub-group of components or a child *concept* in the alternative hierarchy represented by the *Reference set*. The parent of grouping component is represented by the *referencedComponentId*.

Note: Field name in SNOMED CT Release Format 2.

M**mapAdvice (field)**

Field in a *Complex or Extended Map Reference Set* containing human-readable advice, that may be employed by the software vendor to give an end-user advice on selection of the appropriate *target code* from the alternatives presented to him within the group.

mapGroup (field)

Field in a *Complex or Extended Map Reference Set* containing an *integer* that groups a set of complex map records from which one may be selected as a *target code*. Where a *SNOMED CT concept* maps onto 'n' *target codes*, there will be 'n' groups, each containing one or more complex map records.

mapCategoryId (field)

Field in a *Complex or Extended Map Reference Set* that identifies the *SNOMED CT concept* in the metadata hierarchy which represents the *MapCategory* for the associated map member.

 **Note:** The categories vary for different target code systems, each set of categories is represented by a subtype of 609331003|Map category value|. For example in the case of *ICD-10* the individual category values are *subtypes* of:

447634004|ICD-10 Map category value|.

mapPriority (field)



Field in a *Complex or Extended Map Reference Set* that specifies the *order* in which complex map records should be checked. Only the first map record meeting the run - time selection criteria will be taken as the *target code* within each *mapGroup*.

mapRule (field)



Field in a *Complex or Extended Map Reference Set* containing a machine-readable rule, (evaluating to either 'true' or 'false' at run-time) that indicates whether this map record should be selected within its *mapGroup*.

mapTarget (field)



Field in a *Simple Map Reference Set* or a *Complex or Extended Map Reference Set* that contains the *target code(s)* to which the *SNOMED CT concept* represented the *referencedComponentId* is mapped in the *target scheme*.

modifierId (field)



A field in the *relationship file* that indicates the *description logic* modifier that applies to that defining *Relationship* (e.g. "some" or "all").

 **Usage:** Field name in SNOMED CT Release Format 2.

moduleId (field)



A field in each component *release file* which represents the development module within which it was created and is maintained.

 **Note:** Field name in SNOMED CT Release Format 2, which is specified in [see *Identification of Source Module*].

O



order (field)



Order... to be defined.

 **Note:** Field name in SNOMED CT Release Format 2

Q



query (field)



A field in a *Query Reference Set* that contains a text string representing criteria for selection of *SNOMED CT components* to be included in *Simple Reference Set*

 **Note:** A standard syntax for use in these queries is currently under development.

R



referencedComponentId (field)



A field in a *Reference Set* containing an *Identifier* which refers to the *component* to which a row in the *Reference Set* applies.

 **Note:** This field is present in all types of *Reference Set* and, unless otherwise specified, the field data type is *SCTID*.

refsetId (field)



A field in a *Reference Set* which uniquely *Identifier* which refers to the component to which a row in the *Reference Set* applies.

 **Note:** This field is present in all types of *Reference Sets* and its data type is *SCTID*. It links together all the members of a *Reference Set* and refers to a concept that names the *Reference Set*.

Relationship file



The file structure used to distribute *SNOMED CT relationships*.

relationshipGroup (field)



Field in the *Relationship File* is used to group *Relationships* together for a *concept*. For example, where a particular type of prosthesis is inserted a joint, the *Defining characteristics* describing the prosthesis type would be in one group whereas those describing the location or laterality of the joint would be in another group.

S



SCTID (data type)



A unique integer identifier applied to each *SNOMED CT component* (*Concept*, *Description*, *Relationship*).

 **Note:** The value of an SCTID is structured to include an item identifier, a check-digit and a partition identifier. Depending in the value of the partition identifier it may also include a namespace identifier.

sourceEffectiveTime (field)



A field in the Module Dependency *Reference Set* which specifies the *effectiveTime* of the version of the source module with depends on the specified version of the target module. The *effectiveTime* must match exactly.

 **Note:** Field name in SNOMED CT Release Format 2

sourceld (field)



A field in the *Relationship Release File* containing a *SNOMED CT identifier* that refers to the *concept* that represents the source of the associated *Relationship*. The *sourceld* refers to the *concept* that is defined by the *Relationship*.

 **Note:** Field name in *SNOMED CT Release Format 2*. In RF1 this field was called *ConceptId1*

Stated Relationship File



A distribution file containing the *stated form* of *SNOMED CT relationships*.

 **Notes:**

1. The *stated form* of a *Concept* is the *Description Logic* definition that is directly edited by authors or editors. It consists of the stated | is a | *relationships* plus the defining *relationships* that exist prior to running a *classifier* on the logic definitions. Therefore, the *stated form* of a *Concept* is represented by a collection of *relationships*: one or more | Is a | *relationships* and zero or more defining *relationships*.
2. The *Stated Relationships File* is in the same table format as the *Relationships File*, but the value of the *characteristicTypeld* field is | Stated relationship (core metadata concept) |.

Related Links

[Stated Relationships File](#) on page 110

[Stated definition view](#) on page 58

[Inferred definition views](#) on page 58

String (data type)



A datatype representing a sequence of characters.

 **Note:** In *SNOMED CT release file* specifications strings are represented using *Unicode UTF-8* encoding.

T



targetComponentId (field)



An Association Reference Set field containing a *SNOMED CT identifier* which specifies the target of the association from the source component (e.g. a *concept* or *Description*) referred to by the *referencedComponentId*.

 **Note:** Field name in *SNOMED CT Release Format 2*.

targetEffectiveTime (field)



A field in the Module Dependency Reference Set which specifies the *effectiveTime* of the version of the target module on which the specified version of the source module depends. The *effectiveTime* must match exactly.

 **Note:** Field name in *SNOMED CT Release Format 2*

term (field)



A text string that represents the *concept* referenced by the *conceptId* field in the *Description file*.

 **Note:**

By default the *term* is a *UTF-8* string of up to 255 characters. However, description types can be specified which are longer in length and/or contain format markup (e.g. HTML).

Field name in the *Description file*.



Time (data type)

A datatype representing a date or time.

 **Note:** In *SNOMED CT release file* specifications date and times are represented as strings using the ISO 8601 basic format.

- The date format used is YYYYMMDD.
- Where time is included the format is YYYYMMDDThhmssZ. The time is separated from the date by the letter "T" and followed by the letter "Z" indicating that the timezone is UTC.

 **Examples:**

July 31st 2012: **20120731**.

13:15 UTC on August 2nd 2012: **20120802T131500Z**



Transitive closure file

The file used to distribute the *transitive closure* of the *SNOMED CT subtype hierarchy*.

 **Note:** This file is not currently distributed but can be generated from the Relationships file using a script.



typeid (field)

A field in the *Description* and *Relationship Release Files* which contains a *SNOMED CT identifier* that represents the type of *Description* or *Relationship* represented.

- *Description. typeid* represents the type of *Description*. *Description* types include *subtypes* of 90000000000446008 | Description type (core metadata concept) |. These include 90000000000000013009 | Synonym (core metadata concept) | and 9000000000000003001 | Fully specified name (core metadata concept) |. There is no *typeid* value for "Preferred term" as the *preferred term* is the *synonym* marked as "Preferred" in the appropriate [see *Language Reference Set*].
- *Relationship. typeid* represents the type of *Relationship* between the *concept* identified by *sourceld* and the *concept* identified by *destinationId*. *Relationship* types are 116680003 | Is a (attribute) | and *subtypes* of 410662002 | Concept model attribute (attribute) |.

 **Note:** Field name in the *Description file* and in the *Relationship file*.

U



Unicode

A standard character set, which represents most of the characters used in the world using a 16-bit encoding.



 **Note:** The Unicode character set can be encoded using either UTF-16 or UTF-8. UTF-16 uses two bytes for every character. UTF-8 is able to store the most commonly used characters in western alphabets using a single byte, but it requires two bytes to encode accented characters and three bytes to encode symbols used in many non-European scripts.

UTF 16



A standard method of directly encoding *Unicode* using two bytes for every character.

- 👉 **Note:** SNOMED CT release files do not use UTF-16. However, the UTF-8 representation used in release files can be converted to UTF-16.

UTF-8



A standard method of encoding *Unicode* characters in a way optimized for the ASCII character set. *UTF-8* is described in [see [Unicode UTF-8 encoding](#) on page 116].

- 👉 **Note:** This encoding is used for release file fields of data type "String".

UUID (data type)



A datatype representing a sequence of unique *Identifier* encoded as a 128-bit integer.

- 👉 **Note:** In *SNOMED CT release files* UUIDs are represented using as a string following the standard *canonical form*. In this string form a UUID is represented by 32 hexadecimal digits, displayed in five groups separated by hyphens, in the form 8-4-4-4-12 for a total of 36 characters (32 digits and four hyphens).

- 👉 **Example:** ac527bed-9c70-4aad-8fc9-015828b148d9

Alternatives

Universally Unique Identifier
GUID
Globally Unique Identifier

V



valued (field)



Valued... to be defined.

- 👉 **Note:** Field name in SNOMED CT Release Format 2

W



Word equivalents table



A data table in which each row represents a *Word Equivalent*. See [see [Word Equivalents](#) on page 127].

- 👉 **Note:** File or Table name in SNOMED CT toolkit

WordBlockNumber (field)



A field in the *Word Equivalents Table*, which links together several rows which have an identical or similar meaning.

- 👉 **Note:** Field name in SNOMED CT toolkit

WordKey table



A data table relating each word used in SNOMED CT (other than *Excluded Words*) to the *Descriptions*. See [see [Word Search Tables - Summary](#) on page 126].

Note: File or Table name in SNOMED CT toolkit

WordRole (field)



A field in the *Word Equivalents Table*, which specifies the usual usage of this word, abbreviation or phrase, or the usage in which it has a similar meaning to the text in one or more other rows of the table that share a common *WordBlockNumber*.

Note: Field name in SNOMED CT toolkit

WordText (field)



A field in the *Word Equivalents Table*, which contains a word, phrase, acronym or abbreviation that is considered to be similar in meaning to the text in one or more other rows of the table that share a common *WordBlockNumber*.

Note: Field name in SNOMED CT toolkit

WordType (field)



A field in the *Word Equivalents Table*, which specifies whether this row contains a word, phrase, acronym or abbreviation.

Note: Field name in SNOMED CT toolkit

Chapter

12

References



12.1 SNOMED CT Background



12.1.1 SNOMED CT: A Comprehensive terminology for Health Care



SNOMED Clinical Terms (SNOMED CT) was developed between 1999 and 2002 from a convergence of the content of SNOMED Reference Terminology® (SNOMED RT) and *NHS Clinical Terms Version 3 (CTV3)*. This convergence was a result of a strategic alliance between the College of American Pathologists (CAP) and the *UK National Health Service*. In 2007, the *International Health Terminology Standards Development Organisation* acquired *SNOMED CT* and now develops and maintains it on behalf of its *Members* and *Affiliates*.

SNOMED CT combines the robust strength of *SNOMED RT* in the basic sciences, laboratory and specialty medicine with the highly granular clinician focused content of *CTV3* (formerly known as the *Read Codes*). The result is a comprehensive and precise clinical *reference terminology* that provides unsurpassed clinical content and expressivity for clinical documentation and reporting. *SNOMED CT* enables clinicians, researchers and patients to share comparable data worldwide, across medical specialties and sites of care.

SNOMED CT is founded on four basic principles that have guided development of its clinical content and technical design. These principles will continue to guide the evolution of *SNOMED CT* as it adapts and grows in the ever changing global health care environment.

These guiding principles are:

1. Development efforts must encompass broad, inclusive involvement of diverse clinical groups and medical informatics experts;
2. The clinical content must be quality focused and adhere to strict editorial policies;
3. The quality improvement process must be open to public scrutiny and vendor input, to ensure that the terminology is truly useful within healthcare applications;
4. There must be minimal barriers to adoption and use.

The design of *SNOMED CT* has been driven by the expressed needs of software developers for features that improve their ability to develop useful applications. In response to these needs, the design adds unique numeric *Identifiers*, includes links to legacy codes, supports a sustainable migration and maintenance strategy, permits adaptability for national purposes, and fosters alignment with other terminologies and standards such as *HL7*, *LOINC*, and *DICOM*.

The *IHTSDO* believes that *SNOMED CT* delivers a standardized quality clinical terminology that is required for effective collection of clinical data, its retrieval, aggregation and re-use as well as the sharing, linking and exchanging of medical information.

The file format used for distributing *SNOMED CT* content between 2002 and 2011, which is now known as *Release Format 1 (RF1)*, was balloted and approved as an *ANSI* standard. An enhanced file format, *Release Format 2 (RF2)*, has been approved and adopted by the *IHTSDO*. As *RF2* is phased in during 2011, it will simplify change management and add robust facilities for future extensibility.

12.1.1.1 SNOMED CT Quality Development Process



The *SNOMED Clinical Terms* development process incorporates the efforts of a team of internal and external *modelers*. A documented scientific process is followed which focuses on *Understandability, Reproducibility and Usefulness*. Content is defined and reviewed by multiple clinician editors. Conflicts between editors are resolved through an iterative process, based on achieving agreement and consensus, before being entered into the terminology. As necessary, additional experts are consulted to review the scientific integrity of the content.

The integration of *SNOMED RT* and *Clinical Terms Version 3* to create the first release, was a three year process that involved several stages of review and quality assurance:

- *Description mapping*: NHS editors evaluated each *SNOMED CT concept* and *term* and mapped it to the *Clinical Terms Version 3* terminology; *SNOMED CT* editors performed the same task mapping primarily disorders and procedures from *Clinical Terms Version 3* to *SNOMED RT*.
- *Description mapping conflict resolution*: Mapping discrepancies that occurred between NHS and *SNOMED CT* editors underwent a conflict resolution process to definitively place each *concept* within the merged *hierarchy*.
- *Auto-classification*: The merged database following *description mapping conflict resolution* underwent a series of quality control checks including auto-classification to identify and eliminate cycle errors (e.g. *concept A* | is a | *B* and *concept B* | Is a | *A*) and equivalency errors (e.g. where two defined *concepts* have the exact same definition).
- *Hierarchy review*: The reviewed database has undergone auto-classification and further review of inferred hierarchies.
- *Ongoing refinement*: The quality control process is continuously supplemented by feedback from users involved in adoption of *SNOMED Clinical Terms*.

12.1.1.1.1 Extent of Review



The quality processes used in the development of *SNOMED CT* were complemented with external review.

- *Technical review*: The technical specifications for *SNOMED CT* were published for comment on both the *SNOMED CT* and *NHS* websites.
- *Alpha test review*: Forty-two organizations in six countries tested the *SNOMED CT* alpha test file and completed a structured assessment instrument.
- *Alpha test feedback*: Debriefing sessions were conducted in the US, in the UK and in Australia, at which time test sites shared their positive experiences and recommendations for improvement.
- *Peer review*: The methods used in developing *SNOMED CT* were presented in 6 scientific papers at the 2001 American Medical Informatics Association (AMIA) meeting, the largest association of leaders in medical informatics in the world. *SNOMED CT* was also part of an additional three papers and six posters at the 2002 AMIA meeting and additional posters for AMIA 2003 and 2004.

SNOMED CT was also the subject of papers in the American Health Information Management Association (AHIMA) Journal in 2001-2003, posters at 2001 and 2002 annual meetings and presentations at the 2003 and 2004 annual meetings. In addition, AHIMA introduced an education program "Introduction to Clinical terminology" in 2004 which included a *SNOMED CT*.

Early adopters of *SNOMED RT* (a structure that mirrored *SNOMED CT core tables*) were debriefed on their implementation experience in order to identify the key issues to be addressed in the original version of the *SNOMED CT Technical Implementation Guide*.

12.1.1.1.2 Continuous Quality Improvement



Continuous improvement is an aim of the IHTSDO: Updating the breadth and scope of the content to reflect changes in clinical care and advances in medical science; refining the content to deliver greater precision for data collection, retrieval and aggregation; and enhancing the functionality to serve our users better.

12.1.2 Acknowledgments of Contributors to SNOMED CT®



SNOMED CT was originally created by the College of American Pathologists.

SNOMED CT has been created by combining *SNOMED RT* and a computer based nomenclature and classification known as *Clinical Terms Version 3*, formerly known as the *Read Codes Version 3*, which was created on behalf of the U.K. Department of Health and is Crown copyright.

The *IHTSDO* also acknowledges the contributions of:

- The American Academy of Ophthalmology, for the ophthalmology-related portions of this work.
- SNODENT®: the Systematized Nomenclature of Dentistry, copyright 1998, American Dental Association. Used with permission.
- SNOVET®: the Systematized Nomenclature of Veterinary Medicine, copyright 1982, 1993, American Veterinary Medical Association. Used with permission.
- LOINC®: the Logical Observation Identifier Names and Codes, copyright 1995-2008, Regenstrief Institute LOINC Committee. All rights reserved.
- NANDA®: North American Nursing Diagnosis Association Taxonomy II, copyright 2005- 2008, NANDA International. All rights reserved.
- The Perioperative Nursing Data Set® (PNDS), copyright 2002, AORN, Inc. All rights reserved.
- The Omaha System, copyright 1992, Martin and Associates. Used with permission.
- The Clinical Care Classification, copyright 2004, V.K. Saba. Used with permission.
- The Nursing Interventions Classification (NIC), copyright 2004, Mosby, Inc., and the Center for Nursing Classification and Clinical Effectiveness at the University of Iowa College of Nursing. Used with permission.
- The Nursing Outcomes Classification (NOC), copyright 2004, Mosby, Inc., and the Center for Nursing Classification and Clinical Effectiveness at the University of Iowa College of Nursing. Used with permission.
- This work contains material from the AJCC Cancer Staging Manual, Sixth Edition (2002) published by Springer-Verlag New York. Used with permission of the American Joint Committee on Cancer (AJCC), Chicago, Illinois.
- The Anesthesia Patient Safety Foundation's (APSF) Data Dictionary Task Force. Some material contributed. Copyright 2003, APSF, Inc. Used by permission of the APSF.
- This work contains *terms* from the British Association of Dermatology (BAD), and is used by permission of BAD. Crown Copyright 2003 British Association of Dermatologists.
- This work contains *terms* from The Royal College of Anaesthetists (RCoA), and is used by permission of RCoA. Crown Copyright 2003 The Royal College of Anaesthetists.
- This work contains *terms* from the Authorized Osteopathic Thesaurus, and is used by permission of the American Association of Colleges of Osteopathic Medicine and the American Osteopathic Association.

HL7 version 3 - An object-oriented methodology for collaborative standards development



Beeler GW Jr. *HL7 version 3 - an object- oriented methodology for collaborative standards development*. Int J Med Inf. 1998 Feb; 48(1-3): 151-61.

Desiderata for controlled medical vocabularies in the twenty-first century



Cimino JJ in *Methods Inf Med.* 1998 Nov;37(4-5):394-403

This paper identifies some of the key requirements for a clinical terminology. The topics covered include:

- Vocabulary content;
- *Concept* orientation;
- *Concept* permanence;
- Non-semantic *concept identifiers*;
- *Polyhierarchy*;
- Formal definitions;
- Rejection of "not elsewhere classified" terms;
- Multiple granularities;
- Multiple consistent views;
- Context representation;
- Graceful evolution;
- Recognized redundancy.

HL7 Reference Information Model



[RIM] *HL7 Reference Information Model* (www.hl7.org/library/data-model/RIM/modelpage_non.htm)

Quality of clinical information retrieval using a semantic terminological model



Brown PJB, Sonksen P. Evaluation of the quality of information retrieval of clinical findings from a computerized patient database using a semantic terminological model. JAMIA 2000; 7:391-403.

Lexically Assign, Logically Refine strategy for integrating overlapping terminologies



Dolin RH, Huff SM, Rocha RA, Spackman KA, Campbell KE. Evaluation of a "Lexically Assign, Logically Refine" strategy for semi-automated integration of overlapping terminologies. JAMIA 1998; 5(2): 203-13.

Integration of tools for binding archetypes to SNOMED CT



Integration of tools for binding archetypes to *SNOMED CT*, Erik Sundvall, Rahil Qamar, Mikael Nyström, Mattias Forss , Håkan Petersson, Daniel Karlsson, Hans Åhlfeldt and Alan Rector; BMC Medical Informatics and Decision Making 2008, 8(Suppl 1):S7

Normal forms for description logic expressions of clinical concepts in SNOMED RT



Spackman, KA. *Normal forms for description logic expressions of clinical concepts in SNOMED RT*. Proc AMIA Symp. 2001; 627–631;[<http://www.ncbi.nlm.nih.gov/pmc/articles/PMC2243264/>] website.

Representing clinical information using SNOMED CT with different information models



Markwell D, Sato L, Cheetham E: Representing clinical information using *SNOMED Clinical Terms* with different structural information models. [<http://www.kr-med.org/>] website; proceedings of KR-MED May 31/June 2 2008; Phoenix, Arizona, USA 2008.

Toward vocabulary domain specifications for health level 7-coded data elements



Bakken S, Campbell KE, Cimino JJ, Huff SM, Hammond WE. Toward vocabulary domain specifications for health level 7-coded data elements. JAMIA 2000; 7(4): 333-42.

Chapter

13

13 Glossary



This section contains selected terms from the IHTSDO Glossary. The full IHTSDO Glossary is available as follows:

- Online access: www.ihtsdo.org/gl/;
- PDF file (US English): www.ihtsdo.org/gl.pdf;
- PDF file (GB English): www.ihtsdo.org/gl_gb.pdf.

A



Active component



A *SNOMED CT component* that is intended for use. *Release files* contain *Active* and *Inactive components* to provide a historical record of the content of the terminology at different points in time.

 **Note:** A component is active when the most recent row with the relevant Component.id in the *Full Release* of the relevant *Release File* has the value Component.active=1 (one). The most recent row for a component is determined based on the Component.effectiveTime value.

Active concept



A *Concept* that is intended for use. *Release files* contain *Active* and *Inactive components* to provide a historical record of the content of the terminology at different points in time.

 **Note:** A component is active when the most recent row with the relevant Component.id in the *Full Release* of the relevant *Release File* has the value Component.active=1 (one). The most recent row for a component is determined based on the Component.effectiveTime value.

Active description



A *Description* that is intended for use. *Release files* contain *Active* and *Inactive components* to provide a historical record of the content of the terminology at different points in time.

 **Note:** A component is active when the most recent row with the relevant Component.id in the *Full Release* of the relevant *Release File* has the value Component.active=1 (one). The most recent row for a component is determined based on the Component.effectiveTime value.

Affiliate



An *IHTSDO Affiliate Licensee* in accordance with the *IHTSDO Affiliate License Agreement*.

Alternatives

IHTSDO Affiliate
Affiliate Licensee

Affiliate Licence Agreement



The agreement between an *IHTSDO affiliate* (the licensee) and the *IHTSDO* (the licensor) under which developers and implementers are permitted to use the *SNOMED CT International Release* and distribute it to their sub-licensees as part of a software system.

Alternatives

Affiliate Licence

ANSI



American National Standards Institute (ANSI) is a private non-profit organization that oversees the development of voluntary consensus standards for products, services, processes, systems, and personnel in the United States. The organization also coordinates U.S. standards with international standards.

Alternatives

ANSI

American National Standards Institute

Application Programming Interface



Application Programming Interface

A set of rules and specifications that enable communication between software programs. *Application Programming Interfaces* enables interaction between separate software programs, in much the same way that a *user interface* facilitates interaction between humans and computers.

Alternatives

API

Attribute



An *attribute* represents a characteristic of the meaning of a *concept* or the nature of a refinement.

👉 **Note:** An *attribute* has a name which is represented by a *concept*. All the *concepts* that can be used to name *attributes* are *subtypes* of the *concept | concept model attribute*]. An *attribute* is assigned a value (*attribute value pair*) when used in the definition of a *concept* or in a *postcoordinated expression*. The permitted *attribute values (range.)* for an *attribute* depend on the *attribute name* and on the *domain of the concept* being refined.

👉 **Example:** |Finding site|

Alternatives

Concept Model Attribute

Relationship Type

Role

Attribute group



An association between a set of *attribute value* pairs which causes them to be treated separately from other *attribute value* pairs in the same definition or *postcoordinated expression refinement*.

👉 **Example:**

The definition of the *concept |cholecystectomy with exploration of common duct|* has two *|method|* attributes with different values (*|excision -action|* and *|exploration -action|*) and two *|procedure site direct|* attributes with different values (*|common bile duct structure|* and *|gallbladder structure|*). The attributes are grouped so that procedure is not incorrectly classified as an *|excision of common bile duct|*.

Alternatives

AttributeGroup

Attribute name



A concept that represents the type of a *relationship* or the type of a *refinement* in a *postcoordinated expression*. All the *concepts* that can be used to name attributes are *subtypes* of the *concept* | *concept model attribute* |.

Alternatives

Relationship Type
AttributeName

Attribute value



A concept that represents the target of a *relationship* or the value of an *expression refinement* in a *postcoordinated expression*.

Alternatives

Attribute-value
AttributeValue

Attribute value pair



A combination of an *attribute name* and an *attribute value* used to represent a specific type of information in a generic way without altering the underlying structure of an information model. The *attribute name* identifies the type of information and the *attribute value* provides a value.

Note: *Attribute value pairs* are used by *SNOMED CT* in *relationships* and *postcoordinated expressions*. In both cases, the *attribute name* and *attribute value* are expressed using *SNOMED CT concept identifiers*. In the *Relationship file*, the *attribute name* is represented by the *Relationship.typeId* and the *attribute value* by the *Relationship.destinationId*.

Authoritative concept



A concept with a specific meaning defined by an authoritative source such as a national or international professional body or standards organization.

Authorized Triage Organization



An organization approved by the *IHTSDO* to manage and triage change requests to for inclusion of content in the *SNOMED CT International Release* and/or one or more *National Extensions*.

Note: *IHTSDO Members* and their *National Release Centers* are likely to fulfill this role. In addition, *IHTSDO affiliates* and *Standards Development Organizations* may be eligible for consideration as *Authorized Triage Organizations*.

Alternatives

ATO

Automatic classification



A process that generated a logically consistent *subtype classification* by applying *description logic* rules to the stated definitions of a set of *concepts*.

Alternatives

Auto classify

B**Baseline Release****Browser**

A computer application or software tool used for exploring and searching terminology content. A typical *SNOMED CT browser* can locate *concepts* and *descriptions* by *Identifiers* and by searching the text of *description terms*. Various views of located *concepts* may be displayed including the set of related *descriptions*, the hierarchical *relationships* and other defining *relationships*.

Alternatives

SNOMED CT browser

C**Candidate Baseline Release****Canonical form**

An serialized representation of a *SNOMED CT expression* which follows the *normal form* and in which the *refinements*, *attributes* and *attribute groups* are arranged in a standard order.

Cardinality

? A measure of the number of elements in a set. Modeling rules include constraints on the *cardinality* of particular attributes or associations between classes.

CEN

The European Committee for Standardization is a major provider of European Standards and technical specifications. Its mission is to foster the European economy in global trading, the welfare of European citizens and the environment. Through its services it provides a platform for the development of European Standards and other technical specifications.

Alternatives

Comité Européen de Normalisation
European Committee for Standardization
Europäisches Komitee für Normung

CEN TC251

CEN/TC 251 (*CEN Technical Committee 251*) is a committee within the European Committee for Standardization (*CEN*) working on standardization in the field of Health Information and Communications Technology (ICT) in the *European Union*. Its goal is to achieve compatibility and interoperability between independent systems and to enable modularity in *Electronic Health Record* systems.

Check digit

The *check-digit* is the final (rightmost) digit of the *SNOMED CT Identifier (SCTID)*. It can be used to check the validity of *SCTIDs*. *Clinical Information Systems* can use the *check-digit* to identify *SNOMED CT* codes that have been entered incorrectly (typo errors, etc). It is calculated using the Verhoeff algorithm.

Clinical Information System

A computer-based system that is designed for collecting, storing, manipulating and making available clinical information to support the delivery of healthcare services to individual people and populations.

Alternatives

CIS

Clinical Terms Version 3



One of the source terminologies, along with *SNOMED RT*, that were used to develop *SNOMED CT*. *CTV3* is UK Crown Copyright, distributed by the United Kingdom *National Health Service (NHS)*, and is integrated into *SNOMED CT*.

Alternatives

CTV3

Version 3 of the Read Codes

C-NPU



Nomenclature, Properties and Units (C-NPU in collaboration with International Union of Pure and Applied Chemistry (IUPAC) The IFCC-IUPAC coding system Provides a terminology for Properties and Units in the Clinical Laboratory Sciences

Alternatives

Nomenclature, Properties and Units

NPU

IFCC IUPAC

Note: The name of the organization responsible for C-NPU sometimes used as a synonym

Collaborative Space



A web resource with software to help people involved in a common task achieve goals by enabling effective communication within an project or organization.

Note: The *IHTSDO Collaborative Space* supports the communication needs of *IHTSDO* governance and advisory bodies. *IHTSDO* Standing Committees, Affiliate Forum, Member Forum and Working Groups all have *Collaborative Space* Projects each of which contain meeting announcements, discussions, shared documents and issue trackers.

Alternatives

Collabnet

Common Terminology Services 2



An Application Programming Interface (*API*) specification that is intended to describe the basic functionality that needed by healthcare software implementations to query and access terminological content. *CTS2* defines the functional requirements of a set of service interfaces to allow the representation, access, and maintenance of terminology content either locally, or across a federation of *terminology service nodes*.

Note: *CTS2* is specified as an *API* rather than a set of data structures to enable a wide variety of terminological content to be integrated within a common framework without the need for significant migration or rewrite.

Note: *CTS2* was developed from the original the [see [HL7 CTS specification](#)] and is now a joint initiative between HL7 and the [see [Object Management Group \(OMG\)](#)].

Alternatives

CTS2

HL7 CTS2

Complement



In set theory the *complement* of set A relative to the universal set U is the set of all members of U that are not members of A.

Note: Set theory is applied when describing the intended result of combinations of Reference Sets or Constraints.

Component



Refers to any item identified by an *SCTID* in the main body of *SNOMED CT*, or in an authorized *Extension*. The *partition-identifier* indicates the type of component referred to by that *SCTID*. Each *component* is a uniquely identifiable instance of one of the following:

- *Concept*
- *Description*
- *Relationship*

Alternatives

SNOMED CT component

Component history



A record of an addition or change in the *status* of a *SNOMED CT Component* in a particular *Release Version*. Each item of *Component History* is represented by a row in the *Component History Table*.

Compositional grammar



The set of rules that govern the way in which *SNOMED CT expressions* are represented as a plain text string.

Note: The specification of the [see *SNOMED CT Compositional Grammar*] is available as part of the Technical Implementation Guide.

Alternatives

SNOMED CT compositional grammar

Concept



A clinical idea to which a unique *Concept Identifier* has been assigned.

The term *concept* may also be used informally with the following meanings:

- The *concept Identifier*, which is the key of the *Concepts Table* (in this case it is less ambiguous to use the term "conceptId" or "concept code");
- The real-world referent(s) of the *Concept Identifier*, that is, the class of entities in reality that the *Concept Identifier* represents (in this case it is less ambiguous to use the term "meaning" or "code meaning").

Alternatives

SNOMED CT concept

Concept enumeration



Use of *SNOMED CT concept Identifiers* to represent of a set of values for a property of a particular type of *SNOMED CT component*.

Note: The *SNOMED CT concepts* used to represent *concept enumerations* are usually *subtype children* (or *descendants*) of a relevant general *concept* in the *SNOMED CT metadata hierarchy*. Each possible value is represented by a single child *concept*, and the set of values can be used to enable selection from a pick-list of one or more *concepts*.

 **Example:**

- 900000000000446008 | Description type (core metadata concept) |
 - 90000000000003001 | Fully specified name (core metadata concept) |
 - 90000000000013009 | Synonym (core metadata concept) |
 - 900000000000550004 | Definition (core metadata concept) |

Figure 128: Concept enumeration for: Description.typeId



Concept equivalence

Equivalence is the state of two *SNOMED CT concept* codes or *postcoordinated expressions* having the same meaning. *Concept equivalence* can occur when a *postcoordinated expression* has the same meaning as a *precoordinated concept code*; or when two different *postcoordinated expressions* have the same meaning.



Concept Identifier

A *SNOMED CT Identifier* that uniquely identifies a *Concept* (meaning).

 **Example:** For the meaning named | Pneumonia (disorder) |, the *Concept Identifier* is 233604007.



Concept model

A set of rules that determines the permitted sets of *Relationships* between particular types of *concept*. The *Concept Model* specifies the attributes that can be applied to particular *concepts* and the ranges of permitted values for each of these attributes. There are also additional rules on the *cardinality* and grouping of particular types of *Relationships*.

 **Note:** The [see *Concept Model Guide (6)*] (which is part of the Technical Implementation Guide) summarizes the current set of rules applied to modeling *SNOMED CT concepts*. More detailed information, aimed at those involved creating and modeling content, is available in the *SNOMED CT Editorial Guide*.



Constraint

A rule that specifies limits on the attributes, values and associations that may be applied to a particular component.

 **Examples:**

1. A modeling constraint may limit the permissible defining *Relationships* applied to a particular type of *concept*.
2. An instance data constraint may limit the permissible refinements that may be applied to particular *concept*



Context domain

A context domain is a set of values that are, or may be, used in an identifiable logical setting in an application, protocol, *query* or communication specification. A context domain may be very broad (e.g. procedures or diagnoses) or very narrow (e.g. procedures performed by a specialty or possible values for a field in specific message).



Context specific characteristic

A *Relationship* to a target *Concept* that provides information about the source *Concept* that is true at a particular time or within a particular country or organization . Contrast with *Defining characteristic* and *Qualifying characteristic*. Referred to in CTV3 as a 'Fact'.

Context wrapper



The part of a *SNOMED CT expression* that specifies the context that applies to the *focus concept* that it contains.

Example: "Family history of asthma" can be represented by an *expression* in which the *concept* "asthma" is nested within an *context wrapper* that indicates that this is "family history" - rather than a current condition affecting the patient. For further details see [see [Modeling semantic context](#)].

Core file



A distribution file used to represent the main *SNOMED CT components* (*concepts*, *descriptions* and *relationships*).

Note: In the past the term "core" has also been used to refer to the content of the *SNOMED CT International Release* but this usage is deprecated.

Alternatives

SNOMED CT core
Core table
SNOMED CT core table
SNOMED CT core file
Core table

Cross~Mapping



The process of converting data from a representation in one code system, classification or terminology so that it is represented in another code system, classification or terminology.

Note:

The process as a whole includes the preparation and maintenance of resources used to enable this conversion and the application of such resources to convert instance data.

In *SNOMED CT Cross~Mapping* resources are distributed as [see [Simple](#)] and [see [Complex Map Reference Sets](#)]

D



Darwin Information Typing Architecture



The Darwin Information Typing Architecture (*DITA*) is an XML-based architecture for authoring, producing, and delivering information. Although its main applications have so far been in technical publications, *DITA* is also used for other types of documents such as policies and procedures.

Note: *DITA* is used for creation, publication and maintenance of many *IHTSDO* guidance documents.

Alternatives

DITA

Data Analysis System



A computer system that is used to analyze records or other data that is encoded using SNOMED CT, but not if that system is also a *Data Creation System*;

Note: IHTSDO charges fees for use of *Data Analysis Systems* and *Data Creation Systems* in Non-Member Territories.

Data Creation System



A computer system that is used to create records or other data that is encoded using SNOMED CT.

Note: IHTSDO charges fees for use of *Data Analysis Systems* and *Data Creation Systems* in Non-Member Territories.

Data migration



Steps taken to enable legacy data to be accessible as part of a system that uses SNOMED CT.

Note: The objective of *data migration* is to enable data recorded prior to introduction of SNOMED CT can be retrieved and reused within a *SNOMED CT enabled application*. Options for *data migration* include actual conversion of the data or provision of methods for accessing the data in its original form.

Defining relationship



A *relationship* to a target *concept* that is always necessarily true from any instance of the source *concept*.

Example: The *defining relationships* of the concept | gastrectomy | include |method|=|excision - action| and |procedure site - Direct|=|stomach structure|.

Alternatives

Defining characteristic

Delta release



A *Release Type* in which the *release files* contain only component versions created since the previous release. Each component version in a *delta release* represents either a new component or a change to an existing component.

Derivative



A document, subset, crossmap, extension, or other resource that consists of, includes, references or is derived from one or more SNOMED CT *components*.

Alternatives

SNOMED CT derivative

Description



An association between a human-readable phrase (*Term*) and a particular SNOMED CT *concept* code. Each *description* is represented by a separate row in the *Descriptions File*. Each *Description* is assigned a unique *Description Identifier* and connects a *Term* and a *Concept*.

Alternatives

SNOMED CT description

Description Identifier



A SNOMED CT *Identifier* that uniquely identifies a *Description*.

Description logic



A representation of semantic knowledge that allows formal reasoning to be applied based on axioms that state *relationships* between *concepts*.

Note: *Description logic* definitions of *SNOMED CT concepts* are represented by *defining relationships*. The formal rules of *description logic* can be applied to *defining relationships* by software tools (*description logic classifiers*) to interpret the meaning of *concepts*. This enables confirmation of the logical integrity of the terminology, and can also be used to support meaning-based retrieval from *SNOMED CT enabled* record systems.

Alternatives

DL

Related Links

[Wikipedia entry on Description logic](#)

Description logic classifier



A software tool that applies the rules of a *description logic* to a set of data to make inferences about the *relationships* between sets of *concepts*.

Note: *SNOMED CT concepts* and *relationships* are processed by a *description logic classifier* to generate the *subtype hierarchy*. *SNOMED CT expressions* can also be processed by a classifier to make inferences that support selective retrieval.

Alternatives

Classifier

Dialect



A *language* modified by the vocabulary and grammatical conventions applied to the *language* of a particular geographical or cultural environment.

Directed Acyclic Graph



A set of nodes connected to one another by lines (edges) in which each connection has a specified direction such that no route that follows the direction of the connections enters a loop (cycle).

Example: The *SNOMED CT subtype hierarchy* is an example of a *Directed Acyclic Graph*. *SNOMED CT concepts* are nodes and "is a" *Relationships* are the directed lines that connect them. All "is a" *Relationships* lead from a more specific *concept* to a more general *concept*, so a cycle would be a logical error (e.g. if "rubella virus" is a type of "virus" and "virus" is a type of "microorganism", then "microorganism" cannot be a type of "rubella virus").

Alternatives

DAG

Domain



A set of *concepts* which the *Concept Model* permits to be defined or refined using a particular set of *Attributes* and *Ranges*.

Alternatives

Concept model domain

Draft Standard for Trial Use



A *Draft Standard for Trial Use* is a specification and process to allow implementers to test a standard. At the end of the trial period the standard may be balloted, revised or withdrawn.

 **Example:** The joint project between HL7 International and the IHTSDO, *TermInfo*, is an example of an HL7 DSTU.

Alternatives DSTU

Duplicate term



A *Term* that occurs in several *Active Descriptions*. *Duplicate Terms* are valid in *SNOMED CT* since the intention is to provide natural *terms* used by clinicians rather than to apply formalized phraseology. The formalized form is provided by the *Fully Specified Name* and these are not permitted to be duplicated.

Dynamic snapshot view



A "snapshot view" for a specified date that is generated by filtering a "full view".

E



Electronic health record



A systematic collection of health information about individual patients or populations that is stored in a digital form. An *Electronic health record* may contain a complete and detailed record of a patient's health or may consist of a summary of information of particular relevance to continuing delivery of care.

Alternatives EHR

EN13606



Electronic Health Record Communication (EN 13606) European Standard developed by CEN TC251 to define a rigorous and stable information architecture for communicating part or all of the *Electronic Health Record* (*EHR*) of a single subject of care (patient). This is to support the interoperability of systems and components that need to communicate (access, transfer, add or modify) *EHR* data via electronic messages or as distributed objects:

- preserving the original clinical meaning intended by the author;
- reflecting the confidentiality of that data as intended by the author and patient. .

Enabled application



A software application designed to support the use of *SNOMED CT*.

Alternatives

- SNOMED CT enabled application**
- SNOMED enabled application**
- SNOMED CT application**
- SNOMED application**

Enabled implementation



Implementation of information systems that are able to make effective use of *SNOMED CT* in an organization or region.

 **Note:** *SNOMED CT enabled implementation* has a broader meaning than *SNOMED CT enabled application*. An implementation involves practical deployment of one or more applications but extends beyond the software itself to address personnel and organizational issues that allow the potential benefits to be realized.

Alternatives

- SNOMED CT enabled implementation**
- SNOMED enabled implementation**
- SNOMED CT implementation**
- SNOMED implementation**

Equivalence

See *Word Equivalents*, *Phrase equivalence* and *Concept equivalence*.



Expression

A structured combination of one or more *concept identifiers* used to express an instance of a clinical idea.



Note:

An *expression* containing a single *concept identifier* is referred to as a *precoordinated expression*. An *expression* that contains two or more *concept identifiers* is a *postcoordinated expression*.

The *concept identifiers* in a *postcoordinated expression* are related to one another in accordance with rules expressed in the *SNOMED CT Concept Model*.

These rules allow an *expression* to *refine* the meaning of a *concept* by applying more specific values to particular attributes of a more general *concept*.

Example:

284196006 | burn of skin | : 363698007 | finding site | = 33712006 | skin of hand |

Alternatives

- SNOMED CT expression**

Expression refinement



The part of a *SNOMED CT expression* that applies qualifying details to a *focus concept*.

Example: A "spiral fracture of the left humerus" can be represented by an *expression* in which the *concept* "fracture of humerus" if made more specific by the addition of two refinements "laterality: left" and "associated morphology: spiral fracture".

Alternatives

- Refinement**

Extension



A set of terminology *components* and *derivatives* that is created, structured, maintained and distributed in accordance with *SNOMED CT* specification and guidelines within an *extension namespace* allocated by *IHTSDO*. All *Extensions* are dependent on the *SNOMED CT International Edition* and can be used to broaden the scope of coverage and/or to configure the terminology for use in a specific language, specialty or jurisdiction. *Extensions* are created maintained and distributed by *IHTSDO Members* to address specific national, regional and language requirements. *Extensions* are also created, maintained and distributed by *Affiliates* to meet the needs of particular software solutions and customers.

Note: Components in extensions are identified using *extension SCTIDs*. These identifiers include an *extension namespace* which ensures that they do not collide with other *SCTIDs*, and can be traced to an authorized originator.

Alternatives

- SNOMED CT extension**

Extension namespace identifier

See *namespace identifier*.



F



Focus concept



The part of a *SNOMED CT expression* that represents a clinical finding, observation, event or procedure. This *focus concept* may be given context by a surrounding content wrapped and may be made more specific by a refinement.

Example: A past history of replacement of the left hip may be represented by a *SNOMED CT expression* in which the *focus concept* "hip replacement" is refined by "laterality: left" and enclosed in a *context wrapper* representing "past history".

Full release



A *Release Type* in which the *release files* contain every version of every component ever released.

Full view



A view of *SNOMED CT* that includes all the components in a *Full release*. This includes the full history or all components ever released. A *Full view* can be filtered to provide a *Dynamic snapshot* view of the components as they were at any point in the past.

Fully Specified Name



A *term* unique among *active Descriptions* in *SNOMED CT* that names the meaning of a *Concept code* in a manner that is intended to be unambiguous and stable across multiple contexts.

H



Health Level 7



A not-for-profit, ANSI-accredited standards developing organization dedicated to providing a comprehensive framework and related standards for the exchange, integration, sharing, and retrieval of electronic health information that supports clinical practice and the management, delivery and evaluation of health services.

Alternatives

HL7

Health Level 7 Version 3



A standard for communication of health care information developed by HL7. Version 3 is based on a formal development framework and its communication structures are derived as refinements from a *Reference Information Model (HL7 V3 RIM)*.

Alternatives

HL7 V3

Health Level 7 Version 3 Reference Information Model



The *reference information model* on which *HL7 Version 3* is based.

Alternatives

HL7 V3 RIM

Hierarchy



An ordered organization of *concept* codes linked together through `|` is a *relationships*. *Concept* codes linked to their more general parent *concept* codes directly above them in a *hierarchy*. *Concept* codes with more general meanings are usually presented as being at the top of the *hierarchy* and then at each level down the *hierarchy* code meanings become increasingly more specific or specialized. Formally, a *hierarchy* is represented as a *Directed Acyclic Graph*.

HL7 Termino



An *HL7* project that developed the '*HL7 Version 3 Implementation Guide: Using SNOMED CT*' as a *Draft Standard for Trial Use* (DSTU). The purpose of this guide is to ensure that *HL7 Version 3* standards achieve their stated goal of semantic interoperability when used to communicate clinical information that is represented using *concepts* from *SNOMED CT*.

Alternatives

Term Info

I



ICD-10



The International Statistical Classification of Diseases and Related Health Problems 10th Revision (ICD-10) is a coding of diseases and signs, symptoms, abnormal findings, complaints, social circumstances and external causes of injury or diseases, as classified by the *World Health Organization*. (WHO).

ICD-9



The International Statistical Classification of Diseases and Related Health Problems 9th Revision (ICD-9) is a coding of diseases and signs, symptoms, abnormal findings, complaints, social circumstances and external causes of injury or diseases, as classified by the *World Health Organization*. (WHO).

👉 Note: Replaced by *ICD-10*.

ICD-9-CM



"The International Classification of Diseases, 9th Revision, Clinical Modification" (ICD-9-CM), Sixth Edition, issued for use beginning October 1, 2008 for federal fiscal year 2009 (FY09). The ICD-9-CM is maintained jointly by the National Center for Health Statistics (NCHS) and the Centers for Medicare & Medicaid Services (CMS).

IFCC IUPAC



Nomenclature, Properties and Units (C-NPU) in collaboration with International Union of Pure and Applied Chemistry (IUPAC) The *IFCC-IUPAC* coding system Provides a terminology for Properties and Units in the Clinical Laboratory Sciences

Inactive component



A *SNOMED CT component* that is not intended for use. *Active* and *Inactive components* are included in *release files* to provide a historical record of the content of the terminology different points in time.

👉 Note: A component is inactive when the most recent row with the relevant Component.id in the *Full Release* of the relevant *Release File* has the value Component.active=0 (zero). The most recent row for a component is determined based on the Component.effectiveTime value.

Alternatives

Inactive

Inactive concept

A *Concept* that is not intended for use. *Release files* contain *Active* and *Inactive components* to provide a historical record of the content of the terminology at different points in time.

 **Note:** A component is inactive when the most recent row with the relevant Component.id in the *Full Release* of the relevant *Release File* has the value Component.active=0 (one). The most recent row for a component is determined based on the Component.effectiveTime value.

Inactive description

A *Description* that is not intended for use. *Release files* contain *Active* and *Inactive components* to provide a historical record of the content of the terminology at different points in time.

 **Note:** A component is inactive when the most recent row with the relevant Component.id in the *Full Release* of the relevant *Release File* has the value Component.active=0 (one). The most recent row for a component is determined based on the Component.effectiveTime value.

Intellectual property rights

As defined in the IHTSDO affiliate License Agreement: patents, trade marks, service marks, copyright (including rights in computer software), moral rights, database rights, rights in designs, trade secrets, know-how and other *intellectual property rights*, in each case whether registered or unregistered and including applications for registration, and all rights or forms of protection having equivalent or similar effect in any jurisdiction.

 **Note:** The IHTSDO owns the *intellectual property rights* of SNOMED CT. The IHTSDO is responsible for ongoing maintenance, development, quality assurance, and distribution of SNOMED CT.

Alternatives

IPR

Intellectual Property

IP

International edition

The part of the content of SNOMED CT that forms the common foundation to the terminology available to all IHTSDO Members and Affiliates.

 **Notes:**

1. The *International release*, provided by the IHTSDO, may be supplemented by *Extension editions* maintained by IHTSDO Members and Affiliates to meet additional national, local and organizational requirements.
2. See also *International release*, which refers to a release of content from the *International Edition* at a particular release date.

Alternatives

SNOMED CT International release

SNOMED CT International edition

International Health Terminology Standards Development Organisation

The *International Health Terminology Standards Development Organisation (IHTSDO)* is a not-for-profit association that develops and promotes use of *SNOMED CT* to support safe and effective health information exchange.

Alternatives

IHTSDO

Intersection

In set theory the *intersection* of the sets A and B, is the set of all objects that are members of both A and B.

 **Note:** Set theory is applied when describing the intended result of combinations of Reference Sets or Constraints.

International release

The set of *release files* provided on a specified release date, to represent the part of the content of *SNOMED CT* that forms the common foundation to the terminology available to all *IHTSDO Members and Affiliates*.

 **Notes:**

1. The *International release*, provided by the *IHTSDO*, may be supplemented by *Extension releases* provided by *IHTSDO Members and Affiliates* to meet additional national, local and organizational requirements.
2. See also *International edition* which refers to the same general content, without specifying a particular release date.

Alternatives

SNOMED CT International release
SNOMED CT International edition

IS A

The RelationshipType that defines a supertype - *subtype*. *Relationship* between two *Concepts*.

Usually expressed as *subtype* | is a | supertype. For Example, Blister with infection | is a | Infection of skin.

ISO

ISO (International Organization for Standardization) is the world's largest developer and publisher of International Standards. ISO is a network of the national standards institutes from over 160 countries, one member per country, with a Central Secretariat in Geneva, Switzerland, that coordinates the system.

ISO TC215

ISO TC215 is the ISO Technical Committee for Standardization in the field of information for health, and Health Information and Communications Technology (ICT). Its objectives are to enable compatibility and interoperability between independent systems, to ensure compatibility of data for comparative statistical purposes (e.g. classifications), and to reduce duplication of effort and redundancies.

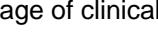
K

Kind of value

The nature of a value that may be associated with a *Concept*. For example, the *concept* | systolic blood pressure | can label a numeric value. The Kind-of-Value that it labels is a pressure.

L**Language**

For purposes of *SNOMED CT* translations, a *language* is a vocabulary and grammatical form that has been allocated an ISO639-1 *language* code. See also *dialect*.

LOINC

Logical Observation Identifiers Names and Codes, a dataset of universal identifiers for identifying medical laboratory observations and other clinical observations to facilitate exchange and storage of clinical results or vital signs.

Alternatives

Logical Observation Identifiers Names and Codes

M**Machine readable concept model**

A representation of the rules that comprise the *SNOMED CT Concept Model* in a form that can be processed by computer software and applied to validate content.

👉 **Note:** The *Machine readable concept model* can be applied to support consistent authoring of *SNOMED CT* content and can also support the creation of valid *postcoordinated expressions* in instance data.

Alternatives

MRCM

Managed content addition

An implementation strategy that involves creating additional *concepts*, *Descriptions* and *Relationships* in an extension so that data can be recorded to the required level of detail using only *precoordinated expressions*.

👉 **Note:** A *description logic classifier* can be used to obtain an updated inferred view of the whole terminology in order to support data retrieval.

Alternatives

MCA

Mapping mechanism

A set of data structures for representing cross-links to other terminologies and classifications. The *Mapping Mechanism* data structures are distributed as three tables:

- *Cross~Map Sets Table*
- *Cross~Maps Table*
- *Cross~Map Targets Table*

Member

A Member of the *International Health Terminology Standards Development Organisation (IHTSDO)* in accordance with the *IHTSDO Articles of Association*.

Alternatives

IHTSDO member**Member territory**

A territory that is represented by an *IHTSDO Member* (as published by the Licensor from time to time)

**Metadata**

SNOMED CT content (including *concepts*, *Descriptions* and *Relationships*) that is used to describe or provide additional information about *SNOMED* content and derivatives (including *reference sets*).

Note:

All *SNOMED CT* metadata *concepts* are *subtypes* of 900000000000441003 | *SNOMED CT Model Component* (*metadata*). The top level of the metadata hierarchy represent broad groups of metadata as shown below.

- 900000000000441003 | *SNOMED CT Model Component* (*metadata*) |
 - 106237007 | *Linkage concept* (*linkage concept*) | ...
 - 370136006 | *Namespace concept* (*namespace concept*) | ...
 - 900000000000442005 | *Core metadata concept* (*core metadata concept*) | ...
 - 900000000000454005 | *Foundation metadata concept* (*foundation metadata concept*) | ...

Figure 129: Top level of the *SNOMED CT* metadata hierarchy

Alternatives***SNOMED CT Metadata*****Migration**

See *Operational migration*, *Data migration* and *Predicate migration*.

**Model of meaning**

An information model that is structured in a way that is designed to provide a common representation of particular types of information which is reusable between different use cases. A model of a meaning combines structural and terminological component in ways that avoid ambiguity and minimize alternative representations of similar meanings.

Example: A model that specifies a how *SNOMED CT* *expressions* are used to represent in a particular *reference information model* to represent clinical findings and procedures in an *electronic health record*.

Note: In contrast, a *model of use* represents the underlying meaning in a way that is determined by a limited set use cases.

**Model of use**

An information model that is structured in a way suggested by a particular intended use of the information that will be represented by that model.

Example: A database that is structured with tables and fields that match specific *user interface* forms and the data entry box on those forms.

Note: In contrast, a *model of meaning* represents the underlying meaning in a way that is common to and reusable between different use cases.

**Modeler**

A person who directly edits the logic definitions and other structures of the terminology. Also sometimes called Clinical Editor or Terminology Manager.

Alternatives

- SNOMED CT modeler**
- Modeller**
- SNOMED CT author**

Modeling



The process of editing logic definitions to reflect the meaning intended by the *Fully Specified Name*.

Alternatives

- SNOMED CT modeling**
- Modelling**
- SNOMED CT authoring**

Monohierarchy



A *Monohierarchy* is a hierarchy in which each node is linked to one and only one parent node.

This type of hierarchy can be represented as a tree with a single root to which each node is attached.

Alternatives

- Monohierarchical classification**

Moved elsewhere



A *Status* value applicable to a *component* that has been moved to another *Namespace*. *Concepts* or *Descriptions* may be moved from an *Extension* to the *International Release*, from the *International Release* to an *Extension* or between one *Extension* and another. Moves occur if responsibility for supporting the *Concepts* changes to another organization .

Note: Component status value

N



Namespace concept



A *Concept* that exists to represent a *SNOMED CT Namespace-Identifier*. All *Namespace Concepts* are direct *subtypes* of the *Concept "Namespace Concept* which is a *subtype* of the *Top-Level Concept "Special Concept"*.

Namespace identifier



A seven digit number allocated by the *IHTSDO* to an organization that is permitted to maintain a *SNOMED CT Extension*. The *namespace identifier* forms part of the *SCTID* allocated every *component* that originated as part of an *Extension*. Therefore, it prevents collision between *SCTIDs* issued by different organizations . The *namespace-identifier* indicates the provenance of each *SNOMED CT component*.

Note: Short format *SCTIDs*,which are used for *components* that originate in the *International Release*, do not include a *namespace-identifier*. In this case the *partition identifier Partition-identifiers* provides sufficient information about the origin of the component.

Alternatives

- Extension namespace identifiers**
- Namespaceld**

National edition



A *SNOMED CT Extension* that is maintained by an *IHTSDO Member* for use in a particular country.

Note: See also *National release*, which refers to a release of content from the *National edition* at a particular release date.

Alternatives

National Extension

National Health Service



Located in the United Kingdom, the *National Health Service (NHS)* worked with the College of American Pathologists in the development of *SNOMED CT*. The *NHS* was one of the founder Members of the *IHTSDO* that is now responsible for *SNOMED CT*.

Alternatives

UK National Health Service

UK NHS

NHS

National Library of Medicine



The *National Library of Medicine (NLM)*, in Bethesda, Maryland, is a part of the National Institutes of Health, US Department of Health and Human Services (HHS). *NLM* is the world's largest medical library. The *NLM* represents the US, as a founder Member of the *IHTSDO*.

Alternatives

NLM

National Release Center



The organization within an *IHTSDO Member* country that is responsible for maintaining and releasing *SNOMED CT* content including any *National Extensions* of *SNOMED CT*.

Natural language processing



Natural Language processing (NLP) is concerned with the interactions between computers and human-readable *languages*. *NLP* includes understanding and generation of human-readable representations. *NLP* understanding systems convert human-readable text into formal representations, which may for example include *SNOMED CT* expressions, to enable more effective processing by other software. *NLP* generation systems convert information from formal representations into human-readable text.

Alternatives

NLP

National release



The set of *release files* provided on a specified release date, to represent the content of a *SNOMED CT Extension* that is maintained by an *IHTSDO Member* for use in a particular country.

Notes:

1. A *National release* adds content to a the *SNOMED CT International Release* for a specified release date, which may be the same as or earlier than the release date of the *National release*.
2. A *National release* is typically maintained and distributed by a *National Release Center*.

Alternatives

SNOMED CT National release

Navigation



The process of locating a *Concept* by traversing *Relationships* or *Navigation links*. For example, moving from a supertype *Concept* to more refined *Concepts*, from a specific *Concept* to a more general *Concept* or from a *Concept* to its *Defining characteristics*. *Navigation Links* allow *navigation* to follow intuitive routes through *SNOMED CT* even where there are no direct supertype or subtype *Relationships*.

Navigation concept



A *Concept* that exists only to support *Navigation*. A *Navigation Concept* is not suitable for recording or aggregating information. All *Navigation Concepts*:

- Are direct *subtypes* of the *concept* "Navigational Concept";
- Have not other supertype or *subtype Relationships*
- Are linked to other *Concepts* only by Navigational Links.

Navigation Hierarchy



A hierarchical view of a set of *SNOMED CT concepts* that is intended to assist navigation at the *user interface*.

Note: There are several differences between *navigation hierarchies* and the formal *subtype hierarchy*:

1. Links between *concepts* in a *navigation hierarchy* are represented by an [see [Ordered Reference Set](#)] (or a [see [Navigation Subset](#)] in *Release Format 1*);
2. *Navigation links* do not contribute to the semantic definitions of *concepts*. Therefore, the criteria for creating a *navigation hierarchy* can be based on arbitrary criteria relating to usability;
3. A *navigation hierarchy* may specify the order in which a set of *concepts* are to be displayed when nested under another specified *concept*.

Non-member territory



A territory that is not an *IHTSDO Member Territory*

Note: In accordance with *IHTSDO affiliate License*, fees are payable to the *IHTSDO* for use of *SNOMED CT* in non-Member Territories.

Normal form



A representation of a *SNOMED CT expression* in which none of the referenced *concepts* are *fully defined* and where there is no redundancy or duplication of meaning.

Notes:

1. *Normal forms* can be used to determine *equivalence* and *subsumption* between *expressions* and thus assist with selective retrieval.
2. Any *SNOMED CT expression* can be transformed to its *normal form* by replacing each reference to a *fully defined concept* with a nested *expression* representing the definition of that *concept*. Transformation rules then resolve redundancies, which may arise from expanding *fully defined concepts*, by removing less specific *attribute values*.

Normal form transformation



The process of converting a *SNOMED CT expression* into its *normal form*.

Notes:

1. The *normal form* provides a way compare different *expressions* which have a similar meaning.
2. The transformation rules are described in [see [Transforming expressions to normal forms](#)].

Alternatives**Transform****Transformation****O****openEHR**

openEHR is an international not-for-profit Foundation working toward making the interoperable, life-long *electronic health record* a reality and improving health care in the information society. It develops specifications that are primarily based on and extend key aspects of the CEN Standard for *Electronic Health Record Communication* (EN 13606).

Operational migration

Steps taken to enable an organization that either used a previous coding scheme (or no clinical coding scheme) to make use of *SNOMED CT*.

P**Partition-identifier**

The second and third digits from the right of the string rendering of the *SCTID*. The value of the *partition-identifier* indicates the type of component that the *SCTID* identifies (e.g. *Concept*, *Description*, *Relationship*, etc) and also indicates whether the *SCTID* contains a *namespace identifier*.

Alternatives**PartitionId****Pending move**

A *Status* value applicable to a *component* that is thought to belong in a different *Namespace* but which is maintained with its *current SCTID* while awaiting addition to the new *Namespace*. A new *Concept* and associated *Descriptions* may be added with this *Status* where a missing *SNOMED CT Concept* is urgently required to support the needs of a particular *Extension*. Existing *Concepts* are also given this *status* when it is recognized that they should be moved to a different *Extension* or to the *International Release*. See also *Moved elsewhere*.

 **Note:** Component status value.

Phrase equivalence

Two words or phrases with a similar meaning. For example, "renal calculus" and "kidney stone".
See *Word Equivalents*.

**Polyhierarchy**

A *Polyhierarchy* is a hierarchy in which each node has one or more parents.

This type of hierarchy can be represented as a graph in which each node has one or more directed links to or from other nodes. Since a node in a hierarchy cannot be a *descendant* of itself the resulting graph must not contain cyclic *Relationships*. This type of graphs is referred to as a "*Directed Acyclic Graph*".

Alternatives**Polyhierarchical classification**

Postcoordinated expression



Representation of a clinical meaning using a combination of two or more *concept identifiers* is referred to as *postcoordination*.

Note: Some clinical meanings may be represented in several different ways. *SNOMED CT* technical specifications include guidance for transforming logical expressions to a common *canonical form*.

Example: *SNOMED CT* includes the following *concepts*:

- 125605004 | fracture of bone |
- 363698007 | finding site |
- 71341001 | bone structure of femur |

SNOMED CT also includes a *precoordinated concept* for 71620000 | fracture of femur |. Therefore It is possible to represent the clinical meaning "fracture of femur" in different ways:

- as a *precoordinated expression*:
 - 71620000 | fracture of femur |
- or as a *postcoordinated expression*:
 - 125605004 | fracture of bone | : 363698007 | finding site | = 71341001 | bone structure of femur |

Alternatives

Postcoordinated
Postcoordination

Precoordinated expression



Representation of a clinical meaning using a single *concept identifier* is referred to as a *precoordinated expression*.

Note: In contrast, *expressions* that contain two or more *concepts Identifier* are referred to as *postcoordinated expressions*. For more information and examples see the glossary entry for *postcoordinated expression*.

Alternatives

precoordinated expression
Precoordinated
Precoordination

Predicate migration



Steps taken to enable pre-existing data retrieval predicates (including queries, standard reports and decision support protocols) to be converted or utilized in a system using *SNOMED CT*.

Preferred term



The *term* that is deemed to be the most clinically appropriate way of expressing a *Concept* in a clinical record. The *Preferred Term* varies according to language and *dialect*.

Note: In *Release Format 2* the *Preferred Term* is indicated by the *typeId* field of a *Language Refset*.

Note: In *Release Format 1* the *Preferred Term* is indicated by a *Language Subset* and/or the *DescriptionType* field of the *Descriptions Table*.

Primitive concept



A *concept* with a formal logic definition that is not sufficient to distinguish its meaning from other similar *concepts*.

Note:

The meaning of *SNOMED CT concept* is expressed in a human-readable form by its *Fully Specified Name*. Each *concept* also has a formal logic definition represented by a set of defining *relationships* to other *concepts*. This logic definition is computer processable. A *primitive concept* does not have sufficient defining *relationships* to computably distinguish them from more general *concepts* (supertypes).

See also *sufficiently defined concept*.

 **Example:** The *concept* 5596004|atypical appendicitis (disorder)| is *primitive* because the following definition is not sufficient to distinguish "atypical appendicitis" from any other type of "appendicitis".

- 116680003 | is a | = 74400008 | appendicitis |
- 116676008 | associated morphology | = 23583003 | inflammation |
- 363698007 | finding site | = 66754008 | appendix structure |

Figure 130: Definition of: |atypical appendicitis (disorder)| (primitive)

Q



Qualifying characteristic



An *attribute-value relationship* associated with a *concept code* to indicate to users that it may be applied to refine the meaning of the code. The set of qualifying *relationships* provide syntactically correct values that can be presented to a user for *postcoordination*. Example: 'Revision status' = 'First revision' is a possible *qualifying characteristic* of 'Hip replacement'. A *qualifying characteristic* is contrasted with a *defining characteristic*. It is referred to in CTV3 as a 'Qualifier'.

Alternatives

Qualifier

Quality characteristic



A type of attribute of a component by which its quality is assessed or measured.

 **Note:** The set of *IHTSDO quality characteristics* are a typology of attributes of an *IHTSDO Component* by which its quality is assessed or measured. A typology is the study or systematic classification of types that have attributes or traits in common.

Quality metric



An agreed method and means for measuring levels of achievement, performance or conformance of a component or its *Quality characteristic(s)*.

Quality target



An agreed level of achievement, performance or conformance of a component for any given *Quality characteristic*.

Query predicate



A statement of a condition that determines whether candidate instance data should be included in or excluded from a selection.

 **Note:** Query predicates applied to a set of *SNOMED CT* expressions may test for subsumption of the overall meaning and/or may test the values applied to particular *attributes* in the expression.

R



Range



A constrained set of values that the *Concept Model* permits to be applied to a specific *Attribute* when that *Attribute* is applied to a *concept* in a particular *Domain*.

Alternatives

Concept model range

Read Code



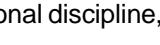
A five-character code allocated to a *concept* or *term* in *CTV3*. Note that codes allocated in *Read Codes Version 2* and the *Read Codes 4-Byte Set* are also included in *CTV3*. The original 4-byte codes are distinguished from 5-byte codes in the general representation by prefixing them with a full stop.

Alternatives

Read Codes 4-Byte Set

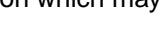
Read Codes Version 2

Realm



A sphere of authority, expertise, or preference that influences the range of *components* required, or the frequency with which they are used. A *Realm* may be a nation, an organization, a professional discipline, a specialty, or an individual user.

Record services



Functions performed by software that interacts with a record system used to capture information which may include references to information in a terminology.

 **Note:** *Record services* are intimately related to ways in which information is entered, stored and retrieved by a particular application. These services interact with *Terminology services* but, unlike *Terminology services* they are usually specific to a particular application.

Reference information model



A high-level generalized model that allows information to be represented and related consistently within a particular field of human endeavor.

 **Note:** The *Health Level 7 Version 3 Reference Information Model* is the most widely used *reference information model* in health care.

Reference set



A work consisting of a set of references to *SNOMED CT* *components* that may associate additional properties with *components* that are members of the set and/or which may indicate associations between members of the set or between members of the set and content of another nomenclature, classification or knowledge structure. The uses of *Reference sets* include identification of subsets of *SNOMED CT* content, representation of alternative hierarchical structures and *Cross-Maps* to classifications.

Alternatives

SNOMED CT reference set

Reset

Reference set member



A uniquely identified reference within a *reference set*.

- 👉 **Note:** Each *reference set* has an identifier (*refsetId*) and contains one or more *reference set members*. Each *reference set member* has its own unique identifier (*id*) which allows it to be versioned using the *effectiveTime* and *active* fields. All *reference set members* also contain a *referencedComponentId* (which refers to a component that is part of the set) and other fields that depend on the type of *reference set*.

Reference terminology



A terminology in which each *term* has a formal computer processable definition that supports meaning based retrieval and aggregation. *SNOMED CT* is a *reference terminology*

Relationship



An association between a source *concept* and a destination *concept*. The nature of the association is indicated by a reference to another *concept* referred to as the *relationship type*.

- 👉 **Notes:**

1. Each *relationship* provides information about the source *concept*. In the example below
2. *Relationships* are represented by rows in the *Relationship File*

- 👉 **Example:**

Table 292: Illustrative example of a Relationship

source	type	destination
74400008 appendicitis	363698007 finding site	66754008 appendix structure

Alternatives

SNOMED CT relationship

SNOMED CT Release



The content of *SNOMED CT* or an internally consistent part of *SNOMED CT* (such as an *Extension*) provided to licensees at a particular point in time.

Alternatives

Release Version
SNOMED CT Edition

Release file



A computer file used to distribute *SNOMED CT* content from the *IHTSDO* (or from the originator of an *Extension*) in a form that can be readily imported by a software application.

SNOMED CT release files follow one of the *release format* specifications *RF1* or *RF2*.

Alternatives

SNOMED CT release file
SNOMED CT distribution file

Release format



A file structure specified by the *IHTSDO* for files used to distribute *SNOMED CT* content.

 **Note:** There are currently two *release formats*: *Release Format 1* and *Release Format 2*.

Alternatives

SNOMED CT release format
SNOMED CT distribution format



Release Format 1

The file structure specified by the *IHTSDO* for the files used to distribute *SNOMED CT* content in 2002.

 **Note:** This format was replaced by *Release Format 2* in January 2012, which is now the primary format for the *SNOMED CT International Release*. However, for backward compatibility *Release Format 1* files can be generated using a conversion utility and continue to be distributed available during an interim transitional period..

Alternatives

SNOMED CT Release Format 1
RF1



Release Format 2

The file structure specified by the *IHTSDO* for files used to distribute *SNOMED CT* content from 2011.

 **Note:** See also: *Release Format 1*.

Alternatives

SNOMED CT Release Format 2
RF2



Release Type

The temporal scope of a *Release Format 2* file or set of files.

Table 293: SNOMED CT Release Types

Release Type	Description
Full	The files representing each type of component contain every version of every component ever released.
Snapshot	The files representing each type of component contain one version of every component released up to the time of the snapshot. The version of each component contained in a snapshot is the most recent version of that component at the time of the snapshot.
Delta	The files representing each type of component contain only component versions created since the previous release. Each component version in a <i>delta release</i> represents either a new component or a change to an existing component.

Root concept



The single *concept* that is at the top of the | SNOMED CT Concept | hierarchy.

Root metadata concept



The single *concept* that is at the top of the | SNOMED CT Model Component (metadata) | hierarchy.

Note: Most of the data in the metadata hierarchy is only relevant to *Release Format 2*. Therefore, this concept may not be present in some *Release Format 1* files.

Alternatives

Root metadata code

S



Situation with explicit context



A *concept* that specifically includes a definition the context of use of a clinical finding or procedure.

Example: "Family history of diabetes mellitus" is a situation with explicit *concept* because it defines the context as "family history". In contrast, "diabetes mellitus" is not a *situation with explicit context* because it can be used in many different situations including "family history", "past medical history", "current diagnosis", etc.

Note: A *situation with explicit context* is defined as a *subtype* of the situation to which it applies with an attribute associating it with the relevant clinical finding or procedure.

Alternatives

Explicit context
Clinical situation

Snapshot release



A *Release Type* in which the *release files* contain one version of every component released up to the time of the snapshot. The version of each component contained in a snapshot is the most recent version of that component at the time of the snapshot.

Snapshot view



A view of *SNOMED CT* that includes all the components in the state there were in at a specified point in time. A *Snapshot view* be provided by a fixed representation that matches the content of a *Snapshot release* or may be generated as a *Dynamic snapshot view* by filtering a *Full view*.

SNOMED



An acronym for the **SystematizedNomenclature of Medicine** originally developed by the College of American Pathologists and now owned and maintained by the *IHTSDO*. *SNOMED Clinical Terms* is the most recent version of this terminology. It was preceded by *SNOMED RT* and *SNOMED International*.

SNOMED Clinical Terms



SNOMED CT is a clinical terminology maintained and distributed by the *IHTSDO*. It is considered to be the most comprehensive, multilingual healthcare terminology in the world. It was created as a result of the merger of *SNOMED RT* and *NHS Clinical Terms Version 3*.

Alternatives

SNOMED CT

SNOMED CT Identifier



A unique *integer identifier* applied to each *SNOMED CT component* (*Concept*, *Description*, *Relationship*, *Subset*, etc.). Each *SCTID* includes an item identifier, a *check-digit*, a *partition identifier* and, depending on the *partition identifier*, may also include a *namespace identifier*.

Alternatives

Identifier

SCTID

SNOMED International



SNOMED International is the version of *SNOMED®* that was first released in 1993 and which, as version 3.5 released in 1998, It was the immediate predecessor of *SNOMED RT*.

SNOMED Reference terminology



The version of *SNOMED®* prior to the collaborative effort to develop *SNOMED Clinical Terms*. It was one of the source terminologies, along with *CTV3*, from which *SNOMED CT* was developed.

Alternatives

SNOMED RT

Sponsored Territory



A Non-Member Territory that has been recognized and designated by the Licensor (*IHTSDO*) as a sponsored territory

👉 **Note:** *SNOMED CT* may be used free of charge by *IHTSDO affiliates* and their sub-licensees in Sponsored Territories. Information about Sponsored Territories is published on the *IHTSDO* web site.

Stated view



The *stated view* of a *Concept* definition consists of the *Relationships* directly edited by terminology authors. It consists of the stated *subtype Relationships* plus the defining *Relationships* that exist prior to running a *Description Logic classifier*.

👉 **Note:** The *Relationships* distributed in the main *Relationships* files are inferred from the stated *Relationships* using a *Description Logic classifier* to ensure consistency and completeness. The *stated view* is distributed in the *Stated Relationships File*.

Alternatives

Stated form

Statistical classification



A hierarchical organization of *terms* or ideas that allows aggregation into categories that can be counted and compared without double counting. A *statistical classification* is monohierarchical which means that each node in the *hierarchy* is part of one node is the level above. This avoids double counting but means that arbitrary decisions must be made where a node is naturally related to more than one parent. For example, in a *statistical classification* such as *ICD-10*, 'bacterial pneumonia' is be related to 'lung disorder' or 'infection disorder' but not to both.

Structure-Entire-Part



A modeling approach used in *SNOMED CT* to represent anatomical entities such as body organs, body systems, body regions,etc.

- **Structure** is the most general way to refer to an organ, body system or region.
- **Entire** refers to a complete organ, body system or region.

- **Part** refers to a part of an organ, body system or region. It explicitly does not refer to the entire organ, body system or region.

 **Example:** The figure below illustrates the relationships between the structure, entire and part concepts applied to the heart.

- 80891009| heart structure |
- 302509004| entire heart |
- 119202000| heart part |

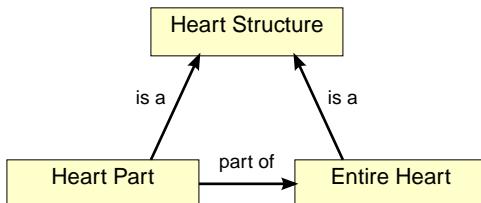


Figure 131: Structure-Entire-Part applied the heart

Alternatives

SEP

Subset



A group of *components* (e.g. *Concepts*, *Descriptions* or *Relationships*) that share a specified common characteristic or common type of characteristic. *Subsets* represent information that affects the way the *components* are displayed or otherwise accessible within a particular *realm*, specialty, application or context.

Subsumption test



A test to determine whether a specified candidate *concept* or *expression* is a *subtype descendant* of another specified *concept* or *expression*.

Alternatives

Subtype test

Subtype



A specialization of a *concept*, sharing all the definitional attributes of the parent *concept*, with additional *defining characteristics*. For example, bacterial infectious disease is a *subtype* of infectious disease. Bacterial septicemia, bacteremia, bacterial peritonitis, etc. are *subtypes* of bacterial infectious disease (and infectious disease as well). *Subtype* is sometimes used to refer to the *concepts* in a *hierarchy* that are directly related to a parent *concept* via the | is a | relationship. In this usage, it is distinguished from *descendants* which explicitly includes *subtypes* of *subtypes*.

Subtype child



A *concept* that has a direct | is a | *subtype Relationship* to a specified *concept*. See also *subtype* and *subtype descendant*.

 **Example:**

The figure below shows an example hierarchy in which concept "E" has three subtype children (G, H and J).

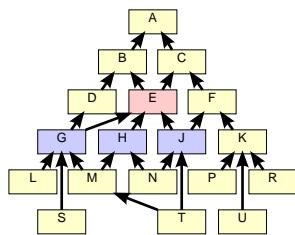


Figure 132: Hierarchy Illustration - Subtype children

Alternatives

Subtype children

Child

Children

Subtype classification

A classification hierarchy in which each node is connected to its supertypes. This allows aggregation of information based on a hierarchy of types.



Alternatives

Subtype hierarchy

Subtype descendant



All subtypes of a concept, including subtypes of subtypes. For example, if a concept has four children, then descendants are those children plus all the concepts that are descended from those four children. See also subtype and subtype child.

Example:

The figure below shows an example hierarchy in which concept "E" has eight subtype descendants (G, H, J, L, M, N, S and T).

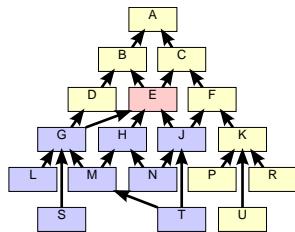


Figure 133: Hierarchy Illustration - Subtype descendants

Alternatives

Descendant

Sufficiently defined concept



A concept with a formal logic definition that is sufficient to distinguish its meaning from other similar concepts.

Note:

The meaning of SNOMED CT concept is expressed in a human-readable form by its *Fully Specified Name* (FSN) and has a formal logic definition represented by a set of defining relationships to other concepts. A *Sufficiently defined concept* has sufficient defining relationships to computably distinguish it from other concepts.

See also *primitive concept*.

Example: The concept 74400008|appendicitis (disorder)| is *sufficiently defined* by the following definition because any *concept* for which this definition was true would be the disorder "appendicitis".

- 116680003 | is a | = 18526009 | disorder of appendix |
- 116680003 | is a | = 302168000 | inflammation of large intestine |
- 116676008 | associated morphology | = 23583003 | inflammation |
- 363698007 | finding site | = 66754008 | appendix structure |

Figure 134: Definition of: |appendicitis (disorder)| (sufficiently defined)

Alternatives

Fully defined concept

Supertype ancestor



Any *concepts* of which the specified *concept* is a *subtype*. Includes the *supertype parents* and the *supertype parents* of each *supertype parent* and so on recursively until the *root concept* is reached.

Example:

The figure below shows an example hierarchy in which concept "T" has ten *supertype ancestors* A, B, C, D, E, F, G, H, J, and M).

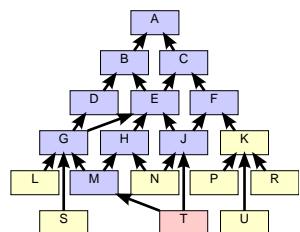


Figure 135: Hierarchy Illustration - Subtype ancestors

Alternatives

Ancestor

Supertype parent



A *concept* that is the target of a direct | is a | *subtype Relationship* from a specified *concept* (see also *supertype ancestor*).

Example:

The figure below shows an example hierarchy in which concept "T" has two *supertype parents* (J and M).

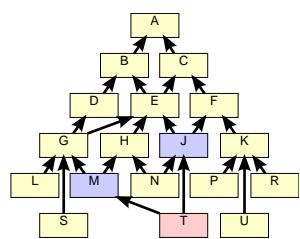


Figure 136: Hierarchy Illustration - Supertype parents

Synonym



A *Term* that is an acceptable alternative to the *Preferred Term* as a way of expressing a *Concept*.

Synonyms allow representations of the various ways a *concept* may be described. *Synonyms* and *Preferred*

Terms (unlike FSNs) are not necessarily unique. More than one *concept* might share the same *Preferred term* or *Synonym*

T



Target code

A code or other *Identifier* within a *Target Scheme*.



Target scheme

A terminology, coding scheme or classification to which some or all *SNOMED CT Concepts* are cross-mapped.



Term

A human-readable phrase that names or describes a *concept*. A *term* is one of the properties of a *description*. Other properties of a *description* link the *term* to an identified *concept* and indicate the type of *description* (e.g. *Fully Specified Name*, *Synonym*, etc.).



Terminology binding



An instance of a link between a terminology component and an information model artifact, such as class or attribute in a *electronic health record* or message.

👉 Notes:

1. Terminology components include *SNOMED CT expressions*, *reference sets* and constraints.
2. Information model artifacts include classes and attributes in reference models for *electronic health records* and communication specifications.
3. *Terminology binding* can also be used to refer to the process of creating and persisting links between terminology components and information model artifacts.

👉 Examples:

1. A set of coded values that may be applied to a particular attribute in an information model. The set may be expressed either explicitly (extensionally) or as a definitional constraint (intensionally).
2. The association between a named attribute value in the information model and a specific coded value or *expression*.
3. A rule that determines the way that a coded *expression* is constructed based on multiple attribute values in the information model.

Terminology server



Software that provides access to *SNOMED CT* (and/or to other terminologies). A *terminology server* typically supports searches and *Navigation* through *Concepts*. A server may provide a *user interface* (e.g. a *browser* or set of screen controls) or may provide low-level software services to support access to the terminology by other applications. See the *SNOMED CT Technical Implementation Guide*.

Alternatives

SNOMED CT terminology server

Terminology services



Functions performed by software that interacts with one or more representations of the terminology and provide access to information derived from the terminology.

 **Note:** Terminology services can be generalized, so that they are independent of the way the terminology is used in a particular application. Terminology services may be used by record services that enter, store and retrieve information that includes SNOMED CT expressions. In contrast to terminology services, record services are usually specific to the design of a particular application.

Textual definition



An additional textual *Description* applied to some SNOMED CT concepts.

 **Note:**

Textual definitions are distributed in a file that follows the same structure as the *Description file* (RF2) but the terms permitted by the "textual definition" are much longer the 255 character limit applied to *synonyms* and *fully specified names*. Textual definitions are not essential for *SNOMED CT implementations* but they are useful as they provide narrative *Descriptions* of *concepts* that may be easier to understand than the shorter terms.

These *Descriptions* go beyond the detail of the *Fully Specified Name* as shown in the example below.

 **Example:**

Table 294: Textual Definition

conceptId	Fully Specified Name	Textual Definition
11530004	Brittle diabetes mellitus (finding)	Diabetes mellitus in which there are frequent, clinically significant fluctuations in blood glucose levels both above and below levels expected to be achieved by available therapies.

Top level concept code



A Concept Code that is directly related to the Root Concept Code by a single Relationship of the Relationship Type "Is a". All Concept Codes (except for metadata concepts) are descended from at least one Top-Level Concept Code via at least one series of Relationships of the Relationship Type "Is a".

Top level metadata code



A Concept Code that is directly related to the Root Metadata Code by a single Relationship of the Relationship Type "Is a". All Metadata Concept Codes are descended from at least one Top-Level Metadata Concept Code via at least one series of Relationships of the Relationship Type "Is a".

 **Note:** Most of the data in the metadata hierarchy is only relevant to Release Format 2. Therefore, this concept may not be present in Release Format 1 files.

Transitive closure



A comprehensive view of all the supertype ancestors of a concept derived by traversing all the "Is a" relationships between that concept and the root concept.

 **Note:** A transitive closure table represents the transitive closure of all active concepts.

Translation



The process of rendering text originally written in one language (source language) into another language (target language).

Translation source language



The language in which the original text is written.

👉 **Example:** English is the source language for the *International edition* of SNOMED CT.

Alternatives

Source language

Translation target language



A language into which the original text is being translated or rendered.

👉 **Example:** For the Spanish language edition, Spanish is the target language.

Alternatives

Target language

Translation Service Provider



Person or organization supplying a translation service.

Alternatives

TSP

U



Understandability, Reproducibility and Usefulness



Criteria applied to test the validity of new *concepts* and design features of SNOMED CT.

- Understandable: The meaning of a *concept* can be understood by an average health care provider, without reference to private or inaccessible information.
- Reproducible: Multiple users apply the *concept* to the same situations.
- Useful: The *concept* has a practical value to users that is self-evident or can be readily explained.

Alternatives

URU

Union



In set theory *union* of the sets A and B, is the set of all objects that are a member of A, or B, or both.

👉 **Note:** Set theory is applied when describing the intended result of combinations of Reference Sets or Constraints.

User interface



The way a software application presents itself to a user including, its on screen appearance, the commands it puts at a users disposal, and the manner in which the user can access and update information by using the application.

Alternatives

UI

V**Value Set**

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

Notes:

1. This definition is used in *HL7 Vocabulary Working Group* documents. In *SNOMED CT* a concept representation may be a *concept identifier* or a *SNOMED CT Expression*.
2. A *Reference set* can be used to represent a value set of *SNOMED CT concepts* each of which is represented by a *concept identifier* in the *referencedComponentId* field.

W**Word equivalent**

A word or abbreviation that is stated to be equivalent to one or more other words, phrases or abbreviations for the purposes of textual searches of *SNOMED CT*. *Word Equivalents* and *Phrase equivalents* are represented as rows in the *Word Equivalents Table*.

Workbench

A set of *IHTSDO* sponsored software tools designed to support the development, maintenance, and use of *SNOMED CT* in health systems around the world.

Alternatives

IHTSDO Workbench

World Health Organization

the directing and coordinating authority for health within the United Nations system. The *World Health Organization (WHO)* maintains the *International Statistical Classification of Diseases and Related Health Problems (ICD)*.

Alternatives

WHO