INTERNATIONAL HEALTH TERMINOLOGY
STANDARDS DEVELOPMENT ORGANISATION

SNOMED CT
Release Format 2.0
Data Structures Specification

Date 20100706
Version 1.0

Amendment History

Version	Date	Editor	Comments
0.1	20090610	John Gutai (IHTSDO)	Draft for internal comment
0.2	20090605	John Gutai (IHTSDO)	First draft for review by Tech Committee
0.3	20090630	John Gutai (IHTSDO)	New format
0.4	20090709	John Gutai (IHTSDO)	Updated with comments
0.5	20090724	John Gutai	Included comments from Community of Practice Review
0.6	20090821	John Gutai	Updated to DFTU, with comments from Member and Committee reviews and QA.
0.7	20090911	John Gutai	Added additional references to metadata
0.8	20100330	John Gutai	Added comments following the Draft for Trial Use period
0.9	20100521	John Gutai	Updated following post DFTU review
0.10	20100521	John Gutai	Updated following final QA review
0.11	20100529	John Gutai	Final feedback from the Technical Committee and UKTC
1.0	20100706	John Gutai	Approved by Management Board as an IHTSDO Standard

Associated Documentation

Document	Location
SNOMED CT File Naming Convention - 090623	Collaborative site
SNOMED CT Release Format 2 – Reference Set Specifications	Collaborative site
SNOMED CT Release Format 2 – Upgrade Guide	Collaborative site

Review Timetable

Review date	Responsible owner	Comments
20120631	CTO	Review after 24 month period

Table of Contents

1 Introduction

1.1 Purpose

This document 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 by the International Health Terminology Standards Development Organisation (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 the National E-Health Transition Authority (NEHTA) in coordination with their Australian Affiliates

Please note that a glossary is not provided within this document. Instead, the glossary on the IHTSDO website should be referenced when that becomes available.

1.2 Who should read this specification?

The intended audience for this document 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.

1.3 Associated Quality Measures

The definition of quality measures to monitor the implementation of this standard do not fall under the scope of this document, but will be published as part of the documentation covering the QA and Release process for the IHTSDO Workbench.

2 General

2.1 File Naming and Layout

In RF2, release files are 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:

<FileType>_<ContentType>_<ContentSubType>_<Country|NamespaceIdentifer>_<Date>.<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.

2.2 Field Data Types

The following data types are used in the release files:

Data Type	Description
SctId	A SNOMED CT identifier, between 6 and 18 digits long, as described in Appendix A
UUID	Universally Unique Identifier, 128-bit unsigned integer
Integer	32-bit signed integer
String	UTF-8 text field
Boolean	Boolean value, represented as one of two possible integer values ('1' for true, '0' for false).
Time	For release files, 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 is used. The time zone is UTC, as indicated by the trailing "Z". (e.g. 20080602T223000Z represents 10:30pm June 2 2008 UTC.)

2.3 Metadata and Enumerated Values

Concept enumerations are used across all release files. A concept enumeration simply uses concepts in a metadata hierarchy (described further in Appendix B) to represent an enumerated value set rather than using integer values directly. A concept enumeration therefore uses 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 current 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 the core 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 set specifications" document.

The third and forth sub-hierarchy under |SNOMED CT Model Concept| are the |Linkage| sub-hierarchy, which holds details of relationship types, and the |Namespace| sub-hierarchy, which holds details of Namespaces.

For more information, see Appendix B.

2.4 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 centres 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 organisation. It is the organisation's responsibility to organise the components in each extension that it is responsible for into one or more modules, in a way that best fits its business needs.

Each SNOMED CT component is 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 is updated (in the usual way by appending a row for the component).

Each module is in one and only one extension. Modules do 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.

2.5 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:

Component Type	Active value	Description of behaviour when most recent row representing a component has the specified Active value
Concept	True	• The Concept is intended for active use. • All active Descriptions for which the conceptId refers to this Concept are valid. Visibility of these active Descriptions depends on information contained in applicable RefsetMembers (for example, whether the Description is in a language dialect reference set that is currently enabled in the vendor's system). • All active Relationships of which it is the sourceId or destinationId are applicable.
Concept	False	• The Concept is not intended for active use. However, it remains a valid concept for historical purposes as part of the SNOMED CT commitment to the principle of 'concept permanence'. • Valid Descriptions of the Concept remain active allowing it to be appropriately viewed in human-readable form. • An inactive Concept cannot be the sourceId, destinationId or typeId of an active Relationship.
Description	True	• The Description contains a Term that is a valid description of the Concept referred to by the conceptId. • An active Description may refer to an inactive Concept, in which case the Term provides a valid description of that inactive Concept. Text based searches should (by default) include only active Descriptions that refer to active Concepts.
Description	False	• The Description is not a valid and the associated Term should no longer be regarded as being associated with the Concept referred to by conceptId.
Relationship	True	• The Relationship represents a valid association of the type specified by the typeId, between two Concepts referred to by the sourceId and destinationId. • An inactive Concept cannot be the sourceId, destinationId or typeId of an active Relationship.
Relationship	False	• The Relationship is not valid. An inactive Relationship should be ignored as it does not apply. • This does not necessarily mean that the association indicated by the Relationship does not apply. The Relationship may be inactive because it is redundant and inferable based on other active Relationships. • An inactive Relationship may refer to either active or inactive Components.
Refset member	True	• The Refset member contains valid information applicable to the Component referred to by the referencedComponentId. • The Component referred to by the referencedComponentId may be active or inactive. An active Refset member cannot make an inactive Component active but may provide related information that continues to be relevant (e.g. the reason for inactivation).
Refset member	False	• The Refset member is not valid. An inactive Refset member should be ignored. The information it contains is not applicable to the Component referred to by referencedComponentId.

2.6 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 always remains 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 inactivate 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 inactivated (as described above), and a new row added with a new id, the effectiveTime set 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 is not 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:

Id	**effectiveTime**	**Active**	**moduleId**	**definitionStatusId**
101291009	*20070701*	1	|Module 1|	|Primitive|

In the next release (on 1st January 2008), the concept is moved from |Module 1| to |Module 2|. Because the moduleId field is not immutable, the concept may be updated simply by adding a new record with the same Id.

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|.

Id	**effectiveTime**	**Active**	**moduleId**	**definitionStatusId**
101291009	*20070701*	1	|Module 1|	|Primitive|
101291009	*20080101*	1	|Module 2|	|Primitive|
101291009	*20080701*	1	|Module 2|	|Fully defined|

In the next release (on 1st January 2009), the concept is deactivated:

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 continues 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 is appended to the release file, not individual records showing each separate edit to the released component.

2.7 Release types

Given the RF2's history tracking capability, it is possible to perform a number of different releases of content:

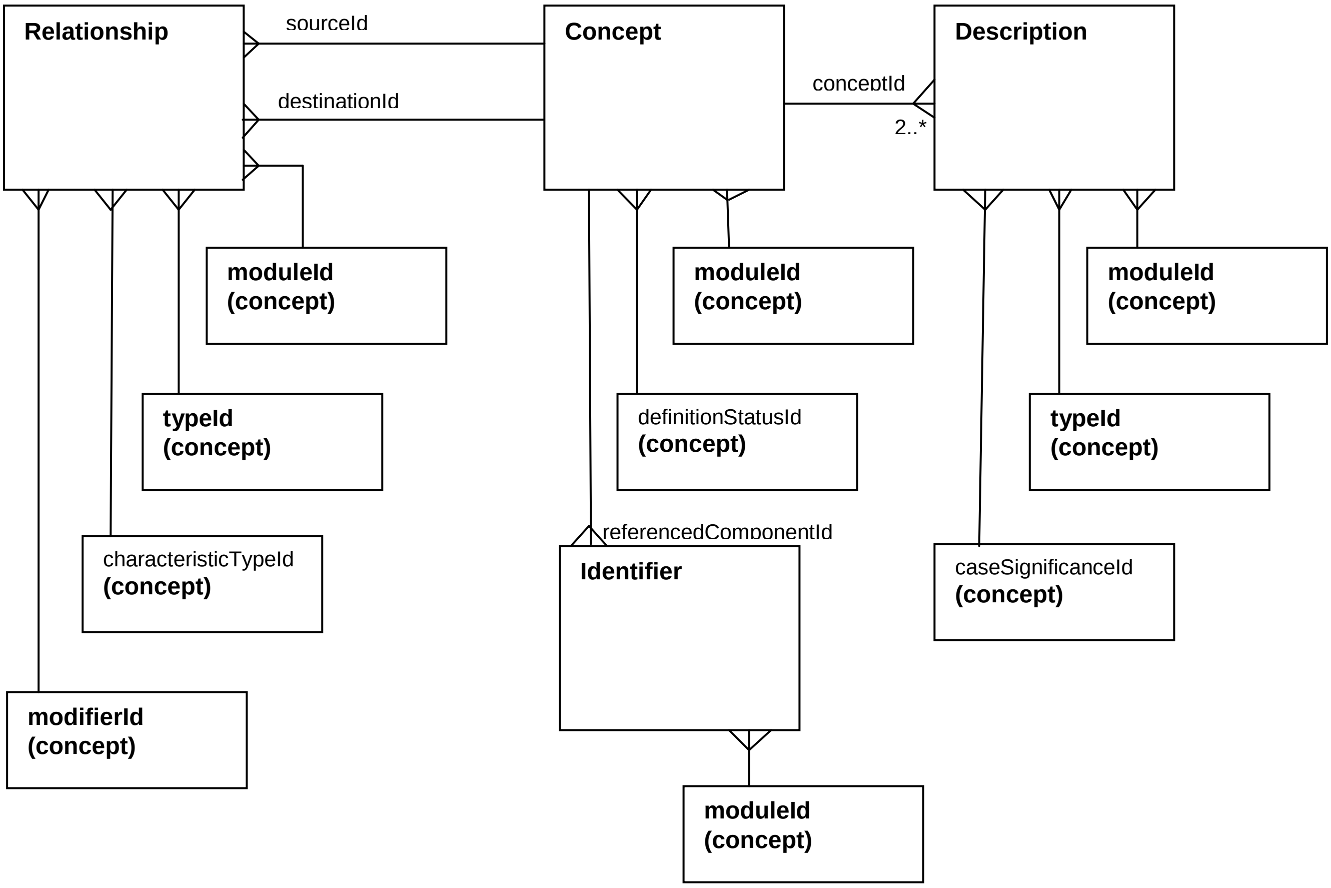
- A "full" release of each file containing every version of every component ever released
- A "snapshot" release, containing only the most recent version of every component ever released (whether this be active or inactive).
- A "delta" release, containing only component versions created since the last release. These component versions may represent a new component or a change to an existing component.

There are of course valid use cases for each type of release form. Each International release incorporates all three of these release types, allowing users to choose the most appropriate format for their needs.

A full release should always be available from release centres. Optionally, other release formats may also be made available. Where out of cycles releases are made, these follow the same format as standard cycle releases.

3 Relationships between files

The relationships between the records in the core files in the RF2 release format are depicted in the following diagram.



Each released version of a |SNOMED CT concept| is held as a single row in the Concept file. Each row represents a clinical concept.

Each released version of a 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 (referenced from the typeId field), 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 child concept connected to each parent concept by an |Is a| relationship. In this hierarchy, each child concept may have more than one parent concept and each parent concept more than one child. All concepts other than the root concept have at least one parent concept. The root of the hierarchy is the |SNOMED CT Concept|, which has a number of top level children, each forming its own sub-hierarchy. There are very few |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.

4 File formats

4.1 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.

Field	Data type	Immutable	Purpose
id	SctId	Y	Uniquely identifies the concept. See Appendix A.
effectiveTime	Time	N	Specifies the inclusive date at which the component version's state became the then current valid state of the component
active	Boolean	N	Specifies whether the concept's state was active or inactive from the nominal release date specified by the effectiveTime
moduleId	SctId	N	Identifies the concept version's module. Set to a descendent of |Module| within the metadata hierarchy.
definitionStatusId	SctId	N	Specifies if the concept version is primitive or full defined. Set to a child of |Definition status| in the metadata hierarchy.

Only one concept record with the same id field is current at any point in time. The current record is 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 retired, 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 are themselves 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 remain unchanged unless incorrect for the concept.
- Rows are 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 retired concept are added to the |Description inactivation indicator reference set|, with an associated value of |Concept non-current|.

4.2 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.

Field	Data type	Immutable	Purpose
id	SctId	Y	Uniquely identifies the description.
effectiveTime	Time	N	Specifies the inclusive date at which the component version's state became the then current valid state of the component
active	Boolean	N	Specifies whether the description's state was active or inactive from the nominal release date specified by the effectiveTime.
moduleId	SctId	N	Identifies the description version's module. Set to a child of |Module| within the metadata hierarchy.
conceptId	SctId	Y	Identifies the concept to which this description belongs. Set to an identifier of a concept in the |SNOMED CT Concept| hierarchy within the Concept file. Note that versions of descriptions and concepts don't belong to each other. Which version of any given description is combined with which version of its owning concept depends on the point in time at which they are accessed.
languageCode	String	Y	Specifies the language of the description text using the two character ISO-639-1 code. Note that this specifies a language level only, not a dialect or country code.
typeId	SctId	Y	Identifies whether the description is an FSN, Synonym or other description

Field	Data type	Immutable	Purpose
			type. This field is set to a child of |Description type| in the Metadata hierarchy.
term	String	N	The description version's text value, represented in UTF-8 encoding.
caseSignificanceId	SctId	N	Identifies the concept enumeration value that represents the case significance of this description version. For example, the term may be completely case sensitive, case insensitive, initial letter case sensitive. This field is set to a child of |Case significance| within the metadata hierarchy.

Only one description record with the same id field is current at any point in time. The current record is 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 do 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 is 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 is appended. Only limited changes may be made to the 'term' field, as defined by editorial rules.

Each concept has 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 is 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) is also configurable for each description type, using the same mechanism as above.
- Control characters (including TABs, CRs and LFs) do not appear in |Plain text| and |Limited HTML| format types.

4.3 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; and
- |Procedure| is the target concept.

Field	**Data type**	**Immutable**	**Purpose**
id	SctId	Y	Uniquely identifies the relationship.
effectiveTime	Time	N	Specifies the inclusive date at which the component version’s state became the then current valid state of the component
active	Boolean	N	Specifies whether the relationship’s state was active or inactive from the nominal release date specified by the effectiveTime field.
moduleId	SctId	N	Identifies the relationship version’s module. Set to a child of |Module| within the metadata hierarchy.
sourceId	SctId	Y	Identifies the source concept of the relationship version, i.e., the concept the relationship version emanates from. Set to an identifier of a concept in the |SNOMED CT Concept| hierarchy within the “Concept” file.
destinationId	SctId	Y	Identifies the concept that is the destination of the relationship version. Set to an identifier of a concept in the |SNOMED CT Concept| hierarchy within the “Concept” file.

Field	Data type	Immutable	Purpose
relationshipGroup	Integer	Y	Groups together relationship versions that are part of a logically associated relationship group. All active Relationship records with the same relationshipGroup number and sourceId are grouped in this way.
typeId	SctId	Y	A concept enumeration value from the metadata hierarchy that identifies the semantic type of the relationship version. For example |Is a|, or |Has associated morphology|.
characteristicTypeId	SctId	Y	A concept enumeration value that identifies the characteristic type of the relationship version (i.e. whether the relationship version is defining, qualifying, etc.) This field is set to a descendant of |Characteristic type| within the metadata hierarchy.
modifierId	SctId	Y	A concept enumeration value that identifies the type of Description Logic (DL) restriction (some, all, etc.). Set to a child of |Modifier| within the metadata hierarchy.

Only one relationship record with the same id field is current at any point in time. The current record is 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 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 do 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 is 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 is 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 is an unsigned integer, and is not limited to a single digit value. There is no guarantee that they are assigned sequentially, and the values are not unique across concepts.

The modifierId field is initially set to |Some| to keep compatibility with the RF1 release. Widening the range of this field to include other values (such as |All|) would 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 would be performed, and the results reviewed by IHTSDO Members and approved by the Technical Committee. The reason that this field has been included at this stage is that as it is core to the semantics of the Description Logic used by SNOMED CT, it should be included as a core field from the start, and not added within the extensibility mechanism at a later date.

4.4 Identifier File

This file provides a standardised way of associating alternate identifiers from various schemes with SNOMED CT components.

At any point in time, an alternate identifier within a particular scheme is 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.

Field	**Data type**	**Immutable**	**Purpose**
identifierSchemeId	SctId	Y	Identifier of the concept enumeration value from the Metadata hierarchy that represents the scheme to which the identifier value belongs. Set to a descendant of |Identifier scheme| within the metadata hierarchy.
alternateIdentifier	String	Y	String representation of the alternate identifier in its native scheme.
effectiveTime	Time	N	Specifies the inclusive date at which the alternate identifier was associated with the SNOMED CT component.
active	Boolean	N	Specifies whether the association was active or inactive from the point in time specified by the effectiveTime.

Field	Data type	Immutable	Purpose
moduleId	SctId	N	Identifies the source module that this association was created in. Set to a child of |Module| within the metadata hierarchy.
referencedComponentId	SctId	Y	Uniquely identifies the SNOMED CT component with which the alternate identifier is associated.

Only one record with the same identifierSchemeId and alternateIdentifier fields is current at any point in time. The current record is 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.

4.5 Transitive Closure History File

The Transitive Closure is the complete set of IS-A 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 is of the following format and contains the valid states of the transitive closure of each concept across all previous releases:

Field	Data type	Purpose
subtypeId	SctId	Id of the concept playing the subtype role. Set to an identifier of a concept in the |SNOMED CT Concept| hierarchy within the “Concept” file.
supertypeId	SctId	Id of the concept playing the super-type role. Set to an identifier of a concept in the |SNOMED CT Concept| hierarchy within the “Concept” file.
effectiveTime	Time	Specifies the inclusive date at which the transitive closure record became valid.

Field	Data type	Purpose
active	Boolean	Specifies whether the identifier version’s state was active or inactive from the point in time specified by the effectiveTime.

5 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.

5.1 The basic reference set member file format

The basic reference set data structure consists of the following fields:

Field	**Data type**	**Immutable**	**Purpose**
id	UUID	Y	A 128 bit unsigned integer, uniquely identifying the reference set member.
effectiveTime	Time	N	Specifies the inclusive date at which this change becomes effective.
active	Boolean	N	Specifies whether the member's state was active or inactive from the nominal release date specified by the effectiveTime field.
moduleId	SctId	N	Identifies the member version's module. Set to a child of |Module| within the metadata hierarchy.
refSetId	SctId	Y	Uniquely identifies the reference set that this extension row is part of. Set to a descendent of |Reference set| within the metadata hierarchy.

Field	Data type	Immutable	Purpose
referencedComponentId	SctId or UUID	Y	Uniquely identifies the component that this row relates to, thus defining membership of this component in the Reference Set. This field can be set to the identifier of a record within the Concept, Description, Relationship or Reference Set member file. However, the content of this field can be further restricted for each reference set by the reference set descriptor (see the "SNOMED CT Release Format 2 - Reference Set Specifications" document for more details).
Zero or more other fields	SctId, String, or Integer	N	Optional field
...	SctId, String, or Integer	N	Optional field

Each reference set is defined as a concept in the metadata hierarchy. There is one active row in the above table for each member of the reference set. Individual reference set members are uniquely identified using a UUID. Each Reference Set member belongs to a single Reference Set (referred to by the refSetId field) and also references 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 is current at any point in time. The current record is 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 do 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 is 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 is appended.

A component may belong to any number of reference sets and to each reference set more than once. In the latter case, there is more than one row with the same refSetId and referencedComponentId, each having different id fields, so co-existing at the same time.

5.2 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 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 depend on a documented set of use cases. The supported patterns may expand over time as further use cases are identified.

6 Appendix A - SctId Format

Components within SNOMED Clinical Terms are identified using numeric identifiers. In the descriptions of the individual tables these identifiers are noted as having the data type SctId. This section describes the characteristics of all fields with the data type SctId.

These identifiers have a common set of characteristics and obey a set of rules, which enable each identifier to refer unambiguously to a single component.

6.1 SctId Data Type

The SctId data type is a 64-bit integer, which is subject to the following constraints:

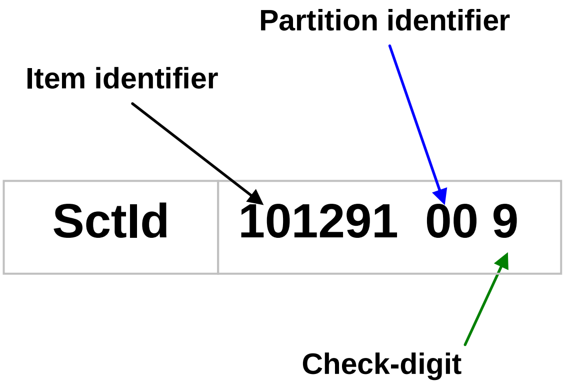
- Only positive integer values are permitted.
- Leading zeros are not permitted.
- The minimum permitted value is 100,000 (6 digits)
- The maximum permitted value is 999,999,999,999,999,999 (18-digits).
- As a result of rules for the partition-identifier and check-digit, many integers within this range are not valid SctIds.

6.2 SctId Representation

The SctId does not contain semantic information related to the meaning of a concept or term. It does however have a structure that is designed to allow different types of terminological components to be recognized. The nature of a component can also be derived from the table in which a component is distributed. However, the advantage of partitioning the SctId is that it avoids reuse of the same identifier for a different type of component – thus avoiding ambiguity. This also allows the nature of the identifier to be recognized when stored in a record or transferred in a message.

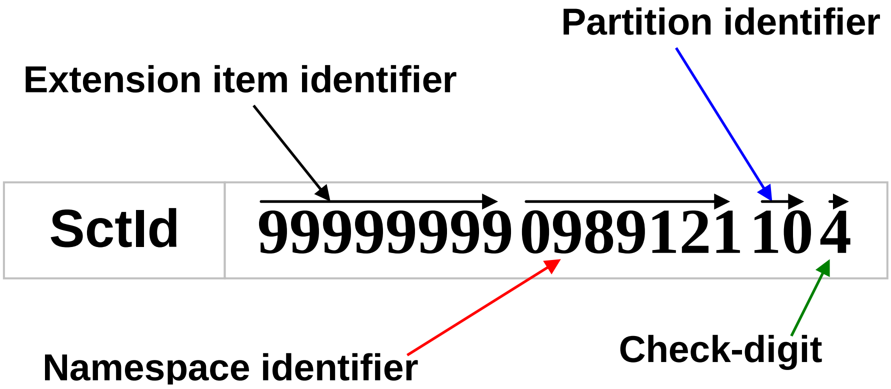
The rightmost digits in a decimal string representation of the SctId have the following defined roles:

- A single check-digit is used to validate the identifier.
- A two-digit partition-identifier, which:
    - Ensures that the identifier is unique in the scope of SNOMED CT; for example, the same identifier cannot be allocated to a Concept and to a Description.
    - Ensures that the identifier of a component in a valid Extension of SNOMED CT cannot be the same as the identifier of a component in the main body of SNOMED CT.



6.3 SctIds and Extensions

If the partition-identifier indicates that the SctId is part of an Extension the next seven-digits (from the right) are a namespace-identifier (see Figure 1). Namespace-identifiers are allocated to organizations which are authorized to issue Extensions. They enable unique SctIds to be issued by many organizations and allow each SctId to be traced to an authorized originating organization.



6.4 SctId Constraints

The constraints on value range for SctIds allow consistent string and integer representation of these values. Exclusion of negative values avoids potential formatting confusion in string representations.

The lower limit of 100,000 ensures the decimal string representation is at least six digits in length. This ensures that an 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.

The upper limit of 18 digits ensures that any valid decimal string can be stored in either a signed or unsigned 64-bit integer.

6.5 Check-digit

The final (units) digit of the SctId is the check-digit. It is not envisaged that users routinely enter 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 (used for the UK NHS number) 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 the *SNOMED Clinical Terms® Technical Reference Guide* for detailed information about the Verhoeff check-digit and sample program code.

6.6 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 entity identified. This allows the identifier of a Description to be distinguished from the identifier of a Concept. It also allows SctIds issued centrally to components of the main body of SNOMED CT to be distinguished from components issued as parts of Extensions.

Identifiers of components in the main body of SNOMED CT have one of the following partition-identifier values:

00	A Concept
01	A Description
02	A Relationship

Identifiers of components in Extensions have one of the following partition-identifier values:

10	A Concept in an Extension
11	A Description in an Extension
12	A Relationship in an Extension

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

6.7 Extension namespaces

If the partition-identifier indicates that the SctId is part of an Extension the seven-digits immediately to the left of the partition-digit are a namespace-identifier.

Each organization that is authorized to issue Extensions is allocated a namespace-identifier. The authorized organization is permitted to assign any valid item identifier that ends with this string of digits.

SNOMED CT core release files include Namespace Concepts representing each of the allocated Namespace-identifiers. These Concepts have the following characteristics:

- They are direct subtypes of the Concept "Namespace Concept" which is a direct subtype of the Concept "Special Concept".
- The Fully Specified Name has the form "Extension Namespace {nnnnnnn} (namespace concept) – where nnnnnnn is the seven digit Namespace-identifier.
- A Synonym associated with the Concept has the form "Extension Namespace nnnnnnn"
- Where appropriate further Synonyms may be included to identify the nature of the Extension and/or responsible organization. However, this information may not be made available for all Namespaces due to privacy constraints.

6.8 Item identifier digits

The remaining digits to the left of the partition-identifier (or in the case of Extensions, to the left of the namespace-identifier) are available to uniquely identify an individual entity within the specified partition. The same item identifier can be allocated in each partition and is rendered unique by the partition-identifier.

For components in the main body of SNOMED CT, item identifiers are usually issued in the arbitrary order in which components are added to SNOMED Clinical Terms. Due to management of the editing process the sequence of issued item identifiers may be discontinuous and the order of identifiers should be regarded as meaningless.

6.9 Example SctIds

The following SctId examples are based on the above rules and illustrate the range of possible item identifiers within each partition.

SctId	**Partition identifier**	**Check digit**	**Notes**
1 0 0 0 0 5	00 =Concept	5 =OK	The Item identifier digits '1000' are the lowest permitted value thus this is the lowest SctId that can be allocated to a Concept.
1 0 0 0 1 4	01=Description	4 =OK	This is the lowest SctId that can be allocated to a Description.
1 0 0 0 2 2	02=Relationship	2 =OK	This is the lowest SctId that can be allocated to a Relationship.
1 0 1 2 9 1 0 0 9	00=Concept	9 =OK	A valid SctId for a Concept.
1 2 9 0 0 2 3 4 0 1 0 1 5	01=Description	5 =OK	A valid SctId for a Description.
9 9 4 0 0 0 0 0 0 1 0 2 9	02=Relationship	9 =OK	A valid SctId for a Relationship.
1 0 0 0 0 0 0 1 1 0 5	10=Extra-Concept	5 =OK	A valid SctId for a Concept in an Extension in the 0000001 namespace.
1 0 9 8 9 1 2 1 1 0 8	10=Extra-Concept	8 =OK	A valid SctId for a Concept in an Extension in the 0989121 namespace.
1 2 9 0 9 8 9 1 2 1 1 0 3	10=Extra-Concept	3 =OK	A valid SctId for a Concept in an Extension in the 0989121 namespace.

SctId	Partition identifier	Check digit	Notes
1290000001117	11=Extra-Description	7 =OK	A valid SctId for a Description in an Extension in the 0000001 namespace.
9940000001126	12=Extra-Relationship	6 =OK	A valid SctId for a Relationship in an Extension in the 0000001 namespace.
999999990989121104			The maximum valid SctId for a Concept in an Extension in the 0989121 namespace.

7 Appendix B – 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 named |SNOMED CT Model Component| is a child of the current root concept |SNOMED CT Concept|, and contain the metadata model, supporting each release.

Underneath the |SNOMED CT Model Component| hierarchy, the |Core metadata concept| sub-hierarchy contains concepts that are referenced from fields within the core 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 set specifications" document.

The third sub-hierarchy under |SNOMED CT Model Concept| is the |Linkage| sub-hierarchy, which holds details of relationship types. The last sub-hierarchy is the |Namespace| hierarchy that hold details of namespaces.

- SNOMED CT Model Component
    - Core metadata concept
        - Description type
            - Definition
            - Fully specified name
            - Synonym
            - Purpose
        - Identifier scheme
            - SNOMED CT UUID
            - SNOMED CT integer ID
        - Modifier
            - All
            - Some
        - Module
            - IHTSDO maintained module
                - SNOMED CT core
                - SNOMED CT model component
        - Case significance
            - Initial character case insensitive
            - Case sensitive
            - Case insensitive
        - Definition status
            - Defined
            - Primitive
        - Characteristic type
            - Additional relationship
            - Defining relationship
                - Inferred relationship

Stated relationship
Qualifying relationship
Foundation metadata concept
Reference set attribute
Reference set
Linkage
Namespace