



SNOMED CT Identifiers

Date 24th May 2011
Version 1.0



Amendment History

Version	Date	Editor	Comments
0.1	2010 11 12	John Gutai	Document created
0.2	2010 12 17	John Gutai	Updated with committee feedback
0.3	2010 12 21	John Gutai	Updated with committee feedback, and separated into separate Specification and Update Guide.
0.4	2010 01 10	John Gutai	Updated following comments from committee review
0.5	2011 04 19	John Gutai	Updated following community of practice review
1.0	2011 05 24	John Gutai	Approved by Management Board

© International Health Terminology Standards Development Organisation 2010. All rights reserved.

SNOMED CT® was originally created by the College of American Pathologists.

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 Licence. Details of the SNOMED CT® Affiliate Licence may be found at <http://www.ihtsdo.org/our-standards/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 Licence. 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from the IHTSDO [or a Member of the IHTSDO] is not controlled by the IHTSDO, and may have been modified and may be out of date.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 <http://www.ihtsdo.org/members/>.



Table of Contents

1 Introduction.....	4
1.1 Purpose of document.....	4
1.2 Scope of document.....	4
2 The SCT Identifier	5
2.1 SctId Data Type	5
2.2 SctId Representation	5
2.3 Short format and Long format SctIds.....	6
2.4 Check-digit.....	7
2.5 Partition-identifier.....	7
2.6 Namespace identifiers	8
2.7 Item identifier digits.....	8
2.8 Example SctIds.....	9
2.9 The Namespace hierarchy.....	10
3 Guidance for Producers of SNOMED CT Identifiers.....	12
3.1 Prerequisites.....	12
3.2 Guidance on Generating SctIds.....	12
3.3 Guidance on Packaging Content.....	12
3.4 Guidance on Promoting Components.....	13
3.5 Guidance on Receiving Promoted Components.....	13
3.6 Guidance on Defaulting Components to their Original Extension.....	14
3.7 Guidance on other Movement of Components between Extensions.....	16
4 Guidance for Consumers of SNOMED CT Identifiers.....	18
4.1 Guidance on validating SctIds within an extension.....	18
4.2 Guidance on identifying the maintaining authority for a component.....	20
4.3 Guidance on parsing and identifying SctIds	20
4.4 Guidance on using state valid data.....	21
5 Guidance where RF1 format is used for an extension.....	22



1 Introduction

1.1 Purpose of document

This document describes the format and usage of SNOMED CT Identifiers, and provides guidance for producers and consumers of SNOMED CT components using SNOMED CT Identifiers.

1.2 Scope of document

This document provides a specification for the format and usage of SNOMED CT identifiers. Please see the separate document “SNOMED CT Identifiers – Update Guide” for a commentary on changes to the existing process and additional information on the benefits of this approach.



2 The SCT Identifier

Components within SNOMED CT are identified using numeric identifiers. In the specifications 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.

2.1 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 six digits.
- The maximum permitted value is eighteen digits.
- As a result of rules for the partition-identifier and check-digit, many integers within this range are not valid SctIds.

2.2 SctId Representation

The SctId does not contain semantic information related to the meaning of a concept or other component. 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 also allows the nature of the identifier to be recognized when stored in a record or transferred in a message.

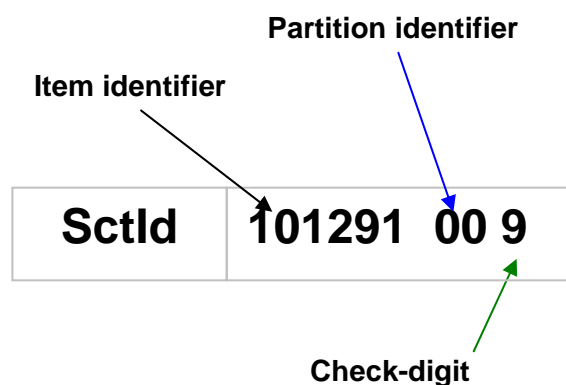
The rightmost digits in the string representation of the SctId have the following defined roles:



- A single check-digit, which is used to validate the identifier.
- A two digit partition-identifier, which can be used to identify the type of component, and also used to separate Sctlds with short formats (that do not include a namespace identifier) from those with long formats (that do include a namespace identifier). Note that only IHTSDO can issue a short format Sctld (without a namespace), whereas an extension owner can issue a long format Sctld (using one of their namespace identifiers).

2.3 Short format and Long format Sctlds

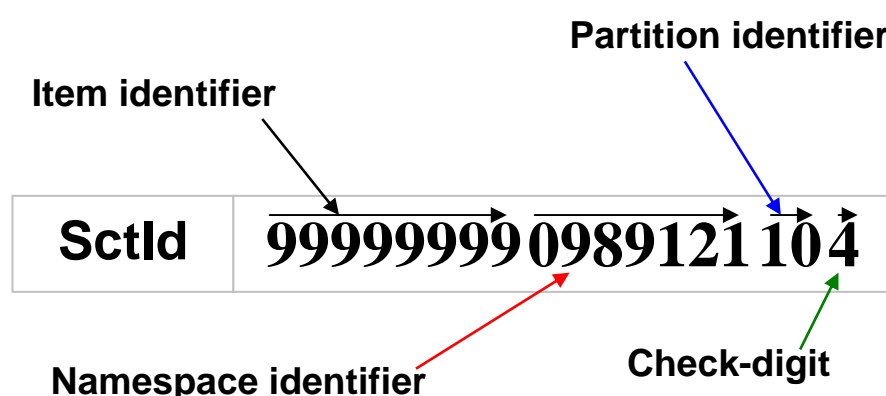
If the partition-identifier indicates that the Sctld has a short format, then the identifier will be made up as follows:



If the partition-identifier indicates that the Sctld has a long format, then the next seven-digits (from the right) are a namespace-identifier. Namespace identifiers are allocated to organizations which are authorized to generate Sctlds. As the namespace identifier is embedded in the Sctld, each



organisation that owns a namespace can generate Sctlds without fear of them clashing with Sctlds that are generated by other organisations. The long format of a Sctld is shown below:



2.4 Check-digit

The rightmost digit of the Sctld is the check-digit. Its purpose is to enable validation of a Sctld to be performed. When a Sctld is generated, the "Verhoeff check" algorithm is used to generate a check-digit (between 0 and 9) from the other digits in the Sctld. In this way, consumers of a Sctld can validate its consistency by regenerating the check-digit using the same algorithm and comparing that with the check-digit embedded in the Sctld.

See the *SNOMED Clinical Terms® Technical Reference Guide* for detailed information about the "Verhoeff check" algorithm and sample program code.

2.5 Partition-identifier

The second and third digits to the right of the Sctld are the partition-identifier. Its purpose is to indicate both the type of component (e.g.: a Description or a Concept) and the format of the identifier (being either of short format or long format).



Short format SctIds for components can have one of the following partition-identifier values:

00	A Concept, using short format
01	A Description, using short format
02	A Relationship, using short format

Long format SctIds for components can have one of the following partition-identifier values:

10	A Concept, using long format
11	A Description, using long format
12	A Relationship, using long format

All other partition-identifier values are invalid.

2.6 Namespace identifiers

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, and described later in this section.

2.7 Item identifier digits

The remaining digits to the left of the partition-identifier (or in the case of long format SctIds, to the left of the namespace-identifier) are the item identifier, used to uniquely identify an individual component. The same item identifier can be allocated across any number of namespaces without any duplication of SctIds, and no meaning should be attached to re-use of item identifiers in this way.



An item identifier can have a lowest permissible value of '100' (three digits) and a highest permissible value of 999999999999999 (15 digits) for short format identifiers or a lowest permissible value of '1' (one digit) and a highest permissible value of 99999999 (8 digits) for long format identifiers or. Leading zeros are not permitted in the item identifier.

Item identifiers may be issued by IHTSDO or by a namespace owner. They are issued in an arbitrary order, usually in the order in which components are added to SNOMED CT. Due to management of the editing process the sequence of issued item identifiers may be discontinuous and the order of these identifiers should be regarded as meaningless.

2.8 Example Sctlds

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

Sctld	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 Sctld that can be allocated to a Concept.
100014	01=Description, using short format	4	This is the lowest Sctld that can be allocated to a Description.
100022	02=Relationship, using short format	2	This is the lowest Sctld that can be allocated to a Relationship.
101291009	00=Concept, using short format	9	A valid Sctld for a Concept.
1290023401015	01=Description, using short format	5	A valid Sctld for a Description.
9940000001029	02=Relationship, using short format	9	A valid Sctld for a Relationship.
10000001105	10=Concept, using long format	5	A valid long format Sctld for a Concept in the 0000001 namespace.
10989121108	10= Concept, using long format	8	A valid long format Sctld for a Concept in the 0989121 namespace.
1290989121103	10= Concept, using long format	3	A valid long format Sctld for a Concept in the 0989121 namespace.
1290000001117	11= Description, using long format	7	A valid long format Sctld for a Description in the 0000001 namespace.



Sctld	Partition identifier	Check digit	Notes
9940000001126	12= Relationship, using long format	6	A valid long format Sctld for a Relationship in the 0000001 namespace.
999999990989121104	10= Concept, using long format	4	The maximum valid Sctld for a Concept in the 0989121 namespace.

2.9 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:

Namespace Concept

- Namespace 1
- Namespace 2
 - Namespace 4
 - Namespace 5
- Namespace 3

In the above sub-hierarchy, |Namespace 1|, |Namespace 2| and |Namespace 3| are all child namespaces of the International edition (which does not have a namespace, and uses short format Sctlds to identify components). Also, |Namespace 2| is the parent namespace of |Namespace 4| and |Namespace 5|. A namespace may only have one parent in the metadata. When requesting a namespace from IHTSDO, there will be a facility to optionally specify a parent namespace.

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 |Namespace Concept| sub-hierarchy. For example, a concept may be moved from the extension in which |Namespace 4| is released to the extension in which |Namespace 2| is released without changing its SCTId. A concept may not be moved from the extension in which |Namespace 4| is released to the extension in which |Namespace



3] is released, unless the original concept is retired and a new SCTId is used. A concept may be moved from any extension to the International release.

Metadata Concepts in this sub-hierarchy have the following characteristics:

- They are subtypes (either children or descendants) of the Concept “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.

To set up a Namespace in the Namespace hierarchy, contact info@ihtsdo.org with details of the Namespace.



3 Guidance for Producers of SNOMED CT Identifiers

3.1 Prerequisites

Before generating SctIds, an organisation must own a namespace. A namespace can be requested from IHTSDO by emailing a request to info@ihtsdo.org.

3.2 Guidance on Generating SctIds

The following guidance is provided for owners of namespaces that generate new content:

- An organisation should only generate new SctIds for components within a namespace that they own.
- An organisation 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.

3.3 Guidance on Packaging Content

Organisations 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 organisation 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 organisation and each group has its own moduleId. Content that is owned by parent organisations may be held in separate files that are also included in the release. Content owned by other organisations 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 organisations may be included in the same release files as content owned by the releasing organisation. 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 organisation, as this would be considered as editing content that the organisation did not own.

3.4 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 organisation 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 inactivate the component). Once the component has been promoted to a parent extension, care should be taken not to amend or inactivate 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.

3.5 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 organisation.



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 organisation.

The `SctId` of the component should not change when it is included in the new extension or the International release.

3.6 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 organisation 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 organisation.

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 inactivate 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.

3.7 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:

- a) To ensure that two components with the same SNOMED CT Identifier are not released independently (and perhaps inconsistently) in two separate extensions.



- b) 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.



4 Guidance for Consumers of SNOMED CT Identifiers

4.1 Guidance on validating Sctlds within an extension

The following checks may be performed to validate the consistency of Sctlds in one or more extensions:

- An extension should only contain components that have a namespace owned by the releasing organisation, or a child of a namespace owned by the releasing organisation. Note however, that a releasing organisation 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 Sctld 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 Sctld 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:

Sctld	effectiveTime	Active	moduleId	definitionStatusId
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:

SctId	effectiveTime	Active	moduleId	definitionStatusId
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	20080131	1	Module 1	Primitive

International release:

SctId	effectiveTime	Active	moduleId	definitionStatusId
1290989121103	20080131	1	Module 2	Primitive



4.2 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 organisation that owns the namespace of the component. If it is not, then the organisation 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.

4.3 Guidance on parsing and identifying SctIds

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.



4.4 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.



5 Guidance where RF1 format is used for an extension

Where a namespace owner is still releasing an extension using RF1 (the old release format) as its master release format, then content should continue to be promoted from and to that extension by creating new SNOMED CT identifiers and using the old “move to / move from” mechanism.

In this case, consumers of RF1 extensions should be aware that content may still have been promoted to the International release or to a parent without a change of SCTId being made, and so may contain components from a number of Namespaces.