REST, CTS2 and URI Mapping

Harold R. Solbrig
Division of Biomedical Statistics and Informatics, Mayo Clinic College of Medicine, Rochester, MN

Abstract. The Common Terminology Services 2 (CTS2) standard provides a normalized representation and set of RESTful access methods for terminological resources. The standard is intended, in part, to serve as a bridge between 'traditional' terminological resources and the Semantic Web. The foundational architecture of CTS2 is based on an architectural "style" known as Resource Oriented Architecture (ROA). One of the requirements of ROA is that every resource must have an identifier. As the Universal Resource Identifier (URI) [30] is the primary identifier used by the Semantic Web, the CTS2 specification calls for all of its resources to be identified by URIs.

This paper discusses some of the ramifications of this decision, including the need to distinguish the identity of the resource being described from the identity of the description itself, and the basic requirement that all servers in a CTS2 'ecosystem' need to use the same identifiers if information is to be aggregated and shared in a meaningful fashion. It evaluates some approaches that have been taken for creating and sharing identifiers, including a recent approach proposed by Miller and others for arriving at well known, shared identifiers in a decentralized environment.

The paper then concludes by describing the behavior of a service that would allow existing identifiers to be gathered, identified and normalized.

1 Introduction

The Common Terminology Services 2 (CTS2) standard [13] provides a normalized representation and set of RESTful [17, 19] [32] access methods for terminological resources. The standard is intended, in part, to serve as a bridge between 'traditional' terminological resources such as ICD-9-CM [20], LOINC [6], the UMLS [27], language codes[29], country codes [23] and the expanding collection of "Ontologies" being developed for the nascent Semantic Web. The foundational architecture of CTS2 is based on an architectural "style" known as Resource Oriented Architecture (ROA) [17][19]. One of the keys to ROA is that every resource must have an identifier. As the Universal Resource Identifier (URI) [30] is the primary identifier used by the Semantic Web, the CTS2 specification calls for all of its resources to be identified by URIs.

This paper discusses some of the ramifications of this decision, including the need to distinguish the identity of the resource being described from the identifier of the description and a basic requirement that all servers in a CTS2 'ecosystem' need to use the same identifiers if information is to be aggregated and shared in a meaningful fashion. It evaluates some approaches that have been taken for creating and sharing identifiers, including a recent approach proposed by Eric Miller [26][16] and others for arriving at well known, shared identifiers in a decentralized environment.

The paper then concludes by describing the behavior of a service that would allow existing identifiers to be gathered, identified and normalized.

2 CTS2

The Common Terminology Services 2 (CTS2) specification Common Terminology Services 2 (CTS2) specification[13] defines a model for "terminological content", which includes information about code systems (ontologies), entities (aka. classes, properties, individuals, concepts, terms, categories), maps between entities, collections of entity references (value sets) and associations between value sets and data element. For each of these resources it defines a collection of API's that allow the read, query, import, export, maintenance and distribution of the resource content. It then maps these models into REST and SOAP platform specific models (PIMs). The information content of these PIM's are represented as XML Schemas, and the REST and SOAP API renderings by WADL [21] or WSDL [12] respectively. When viewed from a strictly technical perspective, this architecture provides sufficient information to allow CTS2 compliant services to be constructed quickly and relatively painlessly. CTS2 authors will quickly discover, however, that there is (at least) one issue that makes the creation of CTS2 documents a difficult process - what to use as URI's.

2.1 Resource Oriented Architecture

CTS2 is founded on a Resource Oriented Architectural (ROA) [17, 19] style. Richardson and Ruby [32] summarize this approach as:

Four concepts:

1. Resources
2. Their names (URIs)
3. Their representations
4. The links between them, and four properties:
    a. Addressability
    b. Statelessness
    c. Connectedness
    d. A uniform interface

The CTS2 specifies each of the above aspects as follows:

Resources: CTS2 is about *terminological resources*. It identifies the following resource types:

- CodeSystem – A "cohesive" set of assertions about one or more Entities
- CodeSystemVersion – The state of a CodeSystem at a given point in time. A "version" or "release".
- Entity – A class, property or individual
- Association – An assertion of a "semantic relationship" between a subject Entity and target Entity, Literal or Expression

- ValueSet – A named collection of rules to create a set of Entity identifiers
- ValueSetDefinition – The state of a ValueSet at a given point in time
- ResolvedValueSet – The result of applying a ValueSetDefinition to one or more CodeSystemVersions. A set of Entity identifiers
- Map – A named collection of rules for mapping one ValueSet onto a second
- MapVersion – The state of a Map at a given point in time
- ConceptDomain – A "Data Element Concept" as defined in ISO 11179-3 [31]
- ConceptDomainBinding – A rule for association a "Data Element Concept" and a ValueSet. A "Data Element" [31]

Their names (URIs): This is the primary subject of this paper and will be discussed in depth in later sections of this paper.

Their representations: The CTS2 specification provides a formal model for each of the resources above. This model is used to create the XML Schema representations of each of the resources in the PIM. Tentatively, the next revision of CTS2 (CTS2 1.1) [15] will include a sanctioned JSON [1] rendering of the same models and we are targeting a "canonical RDF" model based on [34] for CTS2 version 1.2.

The links between them: This is one of the key differentiators between other architectural approaches and ROA. Where other types of architectures such as CTS [33], LexGrid [28] and LexEVS [22] use a containment model to assert relationships, the CTS2 model calls for links. The net result of this approach is that each of the resources described above in 2.1 become stand-alone entities. CodeSystemVersionCatalogEntry is a representation of a CodeSystemVersion resource, and it carries a link to the CodeSystem that it is a version of. Similarly NamedEntityDescription, a representation of the Entity resource carries a link to the CodeSystemVersion from which the description was drawn. These links serve (at least) two purposes:

1. They unambiguously identify the linked resource
2. They provide an identifier that allows the service to de-reference the link, resulting in a CTS2 based description.

Addressability: While not explicitly referenced in Fielding [19], Richardson and Ruby [32] state, "An application is addressable if it exposes the interesting aspects of its data set as resources" [32] (p84). The CTS2 standard does not meet this requirement. There is no requirement in the CTS2 specification itself that the " versionOf " URI of, say, a **CodeSystemVersionCatalogEntry** resource should directly resolve to a description of the code system that it references. This issue, however, is key to much of the discussion that follows, so stay tuned. Also note that the CTS2 specification allows optional hrefs wherever a link URI is provided, where a service can provide an addressable link to the corresponding description.

Statelessness: This is another characteristic that differentiates the CTS2 specification from its predecessors, in that, while the service (obviously) maintains the state of the various resource descriptions that it represents, it cannot expected to maintain any state information between

operations. The Lexicon Query Services (LQS) [24] specification, one of CTS2's early ancestors, used an object-oriented approach. To access an entity description, one first got a handle to a service object, used it to acquire a code system version object, used it in turn to acquire a handle to an entity description object and then used that handle to access specific information. While this could still be implemented in a stateless fashion, typical CORBA implementations maintained all the handle de-reference links on the server, meaning that the client was committed to a single service instance for the duration and that all was potentially lost in the case of a server errors.

While not so egregious, all of the other CTS2 predecessors suffered from this same issue to a greater or lesser degree. The CTS2 standard, however, is designed in such a way that a service never needs to maintain context between resource references.

Connectedness: Resources in an ROA environment should carry links to all their dependents. If a code is used to identify the language of a term, there should be a link to the definition of the code. The same for mime type, predicate, map quality, etc. A collection of resources in an ROA environment should be navigable in the same way that the resources in the archetype ROA environment, the world wide web, are navigable. Users of an ROA implementation should be presented with a virtual "web" of information that will allow them to navigate from the point they are at to any point that it references. CTS2 implements this by incorporating CTS2 in the model definition, by requiring URI's for the references and by providing slots for hrefs wherever they are known.

A uniform interface: CTS2, in part, addresses the notion of a uniform interface by specifying a standard API - the same API that one would use on any service that is CTS2 compliant. In addition to this, however, the CTS2 specification is designed to be mapped directly to the HTTP REST GET, PUT, POST, REMOVE and HEAD methods – generic operations with precisely defined semantics. The imposition of these semantics allow both clients and middleware to know when it is possible to do caching, server replication and load balancing (GET, PUT,REMOVE, HEAD), which operations change the state of the service itself (PUT, POST, REMOVE) and which are not "idempotent" - repeating the operation twice may return a different result than just once (POST). Similar mappings can be made implicit (e.g. SOAP) or explicit depending on the target.

3 Flavors of URI

The CTS2 specification brings a topic of much contention in the W3C community to the forefront—the difference between a description of a resource and the "resource" itself. This discussion is muddled by the fact that, in the W3C community, many resources are digital to begin with— web pages, jpegs, pdfs, etc. that have no "real world" analog. In a rather lengthy discussion thread, Berners- Lee [9] discusses the difference between a URI that describes a car and a URI that references the actual thing. The resolution, in [18], while not addressing the fundamental problem, arrives at a useful solution—at least in a world where URI's are de-referencable. If the GET request for a URI results in an http 2xx response, indicating success, then the URI references a digital resource, and an appropriate digital rendering should be returned. If, however, it returns a 303, then the URI may reference anything. A summary can

be found in [10], “The TAG resolution effectively extends the range of things one can use HTTP URIs. However, it does not allow one to simply serve a web page at a URI which is used for something else. Of course, it is a general principle of web architecture that it is useful to serve information to those that look up a URI. In the case that the URI is not intended to be used for an information resource. ”

In the case of CTS2, this distinction becomes crucial, as there are at least three different resource types in use—the thing being described (e.g. appendicitis), a description of the thing (e.g. SNOMED CT International’s description of “appendicitis”), and a specific representation of a description of a thing (e.g. the “Spackman OWL” rendering of the description of “appendicitis”). For the sake of clarity, we will focus on descriptions of individuals—things that have at least some solid grounding in reality (vs classes ("concepts”) which some schools of philosophy would argue are identical with their descriptions). Take the thing named “France”. This is, in the terms of BFO [3], a fiat object— an arbitrary division of a chunk of land. SNOMED CT includes a description (or “definition” according to SNOMED CT) of France, assigning it the identifier “223666001”, asserting that the definition is “PRIMITIVE”, which means that the definition states that which is necessarily true, given an instance of France but does not include enough information to determine whether a given instance of a West European Country is, indeed, France. The SNOMED CT definition the states that all instances of the class, France, are also instances of the class, West European countries (concept code 223658005).

What is key here is that there are two different things being described here— the country of France itself, and the SNOMED CT “definition” of France. While it might make sense to say, “I intend to visit one of the referents of 223658005 according to the January, 2013 edition of SNOMED CT International”, the sentence “I intend to visit 223658005” makes no sense outside of a surrounding context. This distinction illustrates (a) one of the primary reasons for having URI’s and (b) the need to distinguish the thing being described from the descriptions.

First, the reason for a URI. The statement, “I intend to visit 223658005” makes no sense without a surrounding context. Even if one knows that we are in the context of SNOMED CT, it still is not completely obvious that the number,“223658005” should be interpreted as an SCTID, a count, a (malformed) date or something else entirely. To understand the context, it is necessary to say (or already have agreed upon) something along the lines of “... 223658005 according to the SNOMED CT International concept identifier scheme.” Expressions of this nature, however, are often imprecise, are difficult to construct and are almost impossible to compare in a computational environment. The solution to this is to agree upon a predefined identifier for a “scoping namespace,” which identifies the domain and purpose of an identifier. As an example, the URI of the form, “http://snomed.info/id/” could serve as a pre-agreed identifier which, when concatenated with a “223658005” expresses the same context in a considerably more succinct and computable fashion.

The second issue is that “http://snomed.info/id/223658005” could refer to two different things. The first is the description of 223658005 (according to the SNOMED CT International concept identifier scheme) and the second is the thing being described. Unless one makes this distinction, the question, “Does SNOMED CT correctly define 223658005?” could have two very different meanings— the first being “Is the current SNOMED CT description of 223658005 correctly formatted and meet the

requirements for a SNOMED CT definition?" and the second, "Does the description of 223658005 in SNOMED CT accurately characterize its intended referent?" Note that there are two different things being discussed— the description and the referent of the description. Following a similar line of reasoning to that in the previous paragraph, we could use phrases to provide this distinction—"The SNOMED CT description of the resource http://snomed.info/id/223658005" and "The referent of the resource, http://snomed.info/id/223658005, described in SNOMED CT". As with the earlier example, this presents problems and, as with the earlier example, we can use namespaces to provide a scoping mechanism. This becomes slightly more complex, however, as we now have namespaces scoping namespaces—something along the lines of http://ihtsdo.org/snomedct/description?uri=http://snomed.info/id/223658005 for the description and "http://snomed.info/id/223658005" for the referent.

As mentioned earlier, there is also a third issue. If we actually want a description, there are a myriad of ways to represent it—RF1, RF2, UMLS RRF, OWL 1.1, OWL 2.0, SKOS, HTML, JSON, plain text, ...While some of these representations are directly transformable, some are not— information is lost, as in the case of the SKOS representation or added, as in the case of RRF or OWL. Furthermore, one cannot talk about "the SNOMED CT" description of a given resource without first fixing a version and *which* SNOMED CT (International, US Edition, Australian Edition, etc...) is being used.

3.1 Types of URI in CTS2

The CTS2 specification addresses these issues in a relatively straightforward fashion, by first differentiating the description of a resource from the referent of that description. The Types of URI section in the Core CTS2 [14] model distinguishes PersistentURI from LocalURI, where an instance of a PersistentURI is a reference to the thing being described and LocalURI is the description itself. (The choice of names comes from the fact that the reference itself (e.g. "the referent of 223658005 according to the SNOMED CT International concept identifier scheme.") while not always permanent (France could be conquered and divided...), potentially has an identity beyond the description. The term "LocalURI" refers to the fact that a given rendering of a description can come in multiple syntaxes and formats, be pulled from multiple servers, etc.)

CTS2, not unexpectedly, focuses on one "syntax"— the standard CTS2 representation of terminological resources. A CTS2 description is identified by a LocalURI and the description itself contains a second about URI, which carries a reference to the thing being described. The CTS2 WADL (and equivalent WSDL bindings) provide mechanisms for identifying descriptions within the context of a CTS2 service and for linking different descriptions into the federated "CTS2 ecosystem." This leaves us with an outstanding issue—how do we construct persistent URI's in a consistent way so that it is possible to know that the referent of the SNOMED CT 223658005 in BioPortal [8], ItServer [5], the NCI Thesaurus [7] and HL7 [4] all refer to the same thing?

3.2 Which Persistent URI is the URI?

It is this specific detail that makes CTS2 difficult. As described earlier, the ROA approach used in the CTS2 specification requires referent URI's for pretty much every element that occurs in a CTS2 schema.

The CodeSystemCatalogEntry resource describing SNOMED CT International calls for a URI that references the code system itself, URI's for the distributor, publisher, and other relevant sources, URI's for roles, URI's for the type(s) of resource, etc. It is hard to determine what needs to go into all of these slots. In order to create a CTS2 EntityDescription record one needs to know, at a minimum, the URI's for

1. SNOMED CT International
2. SNOMED CT International, January 2013 Release
3. The reference of the description (France)
4. US English and GB English
5. The various SNOMED CT case settings
6. owl:Class or owl:ObjectProperty

When approached naively, this task is still reasonably simple, as there are many ways to construct a URI, including, but not limited to:

- DCE UUID [2]: Any modern computer has an unlimited supply of unique identifiers at its disposal. Type "uuidgen" on the command line, prefix it with "urn:uuid:" and you have a URI.
- ISO OID [?]: While acquiring a valid OID root is a non-trivial process, once acquired, it can be used to assign a unique identifier to any resource desired. Prefix an oid with "urn:oid:" and you have a URI.
- URI Generators: Sites such as TinyURL (http://tinyurl.com/) allow anything that looks vaguely like a URL to be registered and to be recorded as a shortened URL.
- PURLs: Persistent URL sites (http://purl.oclc.org/docs/index.html, http://www.purlz.org/, http://purl.bioontology.org) provide a way of generating permanent URL's which can be used to provide information and redirect to other URL's.
- LSID Life Science Identifiers: (Despite the momentum behind this initiative in the early 2000's, it is surprisingly difficult to find a citation or reference that is still available) [fill in, including caBIG references].
- DNS URI: The Distinguished Name Service Universal Resource Identifier scheme.

Any of these approaches would work if we weren't faced with an additional requirement: for something like CTS2 to work in a federated world, it is necessary for two or more CTS2 implementations to be able to share information and for each implementation to "understand" what the other is referencing. This means that, not only do I have to generate a URI for SNOMED CT International, but I need to be able to pass it to a completely different service in a way that that service will be able to recognize that my reference and its reference were identical. One way to accomplish this would be to establish a common registrar—a service that we can ask "do these two URI's reference the same thing?" This, however, is rather impractical as, when coming across an external identifier, I would have to iterate over ALL of my identifiers asking the same question repeatedly. Furthermore, every time I created a new URI, I would have to record it with this service and, when I did, how would I identify what it referenced?

A second approach would be to establish the registrar in advance. If a single registrar existed for all CTS2 service instances, I could then ask this shared service "Give me the URI for the SNOMED CT reference to

France." You could do the same and we would have comparable URI's. There are two flaws, however in this system. The first is strictly political—there have been multiple attempts to establish common authorities (ISO Registrar Registry, HL7 OID's, various PURLz, DOI's, identifiers.org, ...). Establishing a common authority, however, requires significant community buy-in, which, in turn involves trying to convince the community that your registrar is better in some way than all the others. The second issue still stands as well—exactly how DO I ask the question, "give me a URI that stands for the SNOMED CT description of 'France'" without already having a URI that already says exactly that?

The second issue— creating a URI that represents the reference to X in coding system Y— can be resolved by publishing a collection of base URI's and an algorithm for constructing an arbitrary reference. As an example, one might say that one constructs a reference to SNOMED CT concept X by concatenating "http://purlz.org/1234/" to X. This solution, or variations thereof, is what is in play today. The problem, however, still lies in establishing a base URI and algorithm. We still have the basic question of who establishes these bases. Today, there are multiple approaches available for many (but not all—see below) terminology resources. (Examples of SNOMED CT).

These approaches exist for at least two reasons. The first was mentioned above—if you want to become the definitive resource you need to (a) push and promote your approach as superior to all others and (b) have sufficient funds that, once you have become dominant, you can continue to back what you have done. To date, this has not worked. The second reason that this hasn't worked is, in the absence of a single "the" resource, how do I know who to ask for the definitive URI and algorithm for constructing SNOMED CT identifiers? In some cases I find all sorts of approaches, and have no idea which is right—which a different CTS2 implementation is using. In other cases I find none at all. What IS the URI for "US English"? Curiously, there is no obvious answer, which means that I fill something in that is pretty much guaranteed not to work with another solution.

An obvious solution to this dilemma would be to ask the publishers of the *actual resources* to tell us which base URI and identification scheme should be used for the resources at their disposal. Doing so eliminates the problem of competing single authorities. Instead of ONE authority for all resources on the web (a very un-web like thing to do), each publisher becomes their own authority. If you want SNOMED CT URI's you ask IHTSDO how to construct them. If you want ICD-9 URI's, you ask the World Health Organization; NCI Thesaurus, you ask the NCI; LOINC, Regenstrief, etc. This approach has already proven successful with the W3C, where its document identifications scheme includes versioned and non-versioned URI's, but the URI's all begin with some variation on "w3c.org"—the DNS owned and sanctioned by the W3C. Similarly, the Dublin Core Initiative has been successful in publishing their reference documents at http://dublincore.org/, and the actual documents contain the various sanctioned (http://purl.org/) based URI's.

This appears to be the most workable approach and efforts are underway to get key organizations such as the WHO, IHTSDO, NLM, NCI and others to adopt them.

This approach still leaves us with some issues we need to resolve, including:

1. How do I find the sanctioned scheme?

2. How do I determine whether a URI I have been given is “official” and, if so, what its official source is?
3. How do I deal with publishers that, for one reason or another, do not publish official URI’s?
4. What do I do with all the different URI’s and other identifiers that are in use today?

We can begin by addressing the second question. Given that we have a URI, we need to be able to answer two questions:

1. Is it valid?
2. Is the source that said it was trusted?

Fortunately, the Internet DNS system provides an easy way to answer to both of these questions. If we construct a URI that uses http URL scheme, one can use the Internet DNS system to de-reference it. If the de-reference works, the resulting web page can tell us, “Yes, this is a valid URI.” As an example, if the web page addressed by the URI, “http://snomed.info/id/223658005” responds with a 200 or 303 code, this indicates that the organization that owns the name “snomed.info” acknowledges “id/223658005” as valid URI and can provide any additional information they so choose. The answer to the second question can be resolved by determining who owns the DNS root, as they control what is on the site it addresses. A whois (http://http://www.internic.net/whois.html for example) registry search shows that “snomed.info” is registered to Ulrich Anderson at IHTSDO. Resolving http://snomed.info redirects to “www.ihtsdo.org,”another approach to validating that the base is valid.

Now that we’ve answered this question, we can go back to the first—how do I find the sanctioned scheme? Now that we have a way of determining *whether* a scheme is THE (or A) sanctioned scheme by the publishing organization, it is no longer necessary to have a single, authoritative reference source. While there would definitely be advantages to having a well known reference source of this information, it is not absolutely required. Multiple sources can exist because the information on them can be verified. Sensibly organized identification schemes will (and should) allow the algorithm to be determined by itself (who or SNOMED CT example).

The leads to another question, however: What of publishers that don’t publish official URI’s? We can begin by subdividing these into two “camps”. The first “camp” are those who, while not officially sanctioned, are widely enough used that there is no question about them or their lineage. Dublin Core, FOAF, SKOS, RDF, RDFS, OWL, the Wine Ontology and many other W3C published resources fit into this camp. It should also be noted that resources that are officially (vs. SNOMED) in RDF and OWL already have these identifiers by default. The second “camp” is the one we need to focus on. How do we address LOINC, ICD-9-CM, and other code systems that lack URI’s today?

We would propose that, in the cases where sanctioned URI’s don’t exist, we refer to a sufficiently authoritatively source— one that can be trusted to tell us (a) there is NOT a sanctioned URI available on the part of the publisher and (b) here is proxy that is sanctioned by this authority. Fortunately, in the medical domain we have just such an authority— the National Library of Medicine. They already have identifiers (VSAB, RSAB and others) and they have a sanctioned, well known URI. (describe scheme).

This leads us to question (4) What to do with all the URI's and other identifiers in place today? A follow-on to this question would be "What do we do with the NLM sanctioned URI if, say, Regenstrief decides to register an official LOINC URI or the CDC decides to piggy-back on the WHO scheme?"

The CTS2 1.0 specification addresses this issue by mapping the PIM read(NameOrURI,...) : (Resource) to the REST signatures:

<service>/<path>/{name}

<service>/<path>/_byuri?uri=...

The first form (e.g. http://service.org/cts2/codesystem/SNOMEDCT) returns a representation of the resource (in this case CodeSystemCatalogEntry) known to the service locally as "SNOMEDCT." The second form, (e.g. http://service.org/cts2/codesystembyuri?uri=http://snomed.info/sct/900000000000207008) results in a 303 redirect to the same resource as the first form. The service should exhibit the same behavior for any known URI (e.g. http://service.org/cts2/codesystembyuri?uri=urn:oid:2.16.840.1.113883.6.96), and the resulting resource should have its *about* attribute set to the URI considered to be 'official' by the providing service and its *alternateID's* set to the alternatives byid?id=" that will accept URI's, well known names, etc. (In CTS2 1.0 this behavior is constrained to known URI's but in CTS2 1.1 a more generic function will be specified that will accept "... {resource).

This, however, is begging the question as CTS2 service implementers still have to know what is known, what is preferred, etc. The next section proposes a generic service that provides the necessary information in a generic way.

4 Mapping Service

This section describes the abstract behavior of an identifier mapping service—a service that provides 'sanctioned' names and URI's for well-known identifiers and that lists all known identifiers for a given 'sanctioned' name. We begin by describing the various data types that are used by the functions in the sections that follow.

4.1 Data Types

A mapping service instance uses the following abstract elements:

- Identifier - an arbitrary identifier that uniquely references a 'resource' (code system, code system version, value set, value set definition, map, source, ...) within the context of that resource type.
- NCName - an identifier that conforms to the NCName construct defined in XML 1.0 NCName construct[11].
- IRI - an identifier that conforms to the absolute-IRI construct in the Internationalized Resource Identifiers specification [25].

```
Type              ::= 'ASSOCIATION' |
                      'BINDING_QUALIFIER' |
                      ... (ReferenceTypes from CTS2 Core document)
Source            ::= STRING
Description       ::= STRING
SourceDescription ::= Source [Description]
Identifier        ::= STRING
NCName            ::= STRING (CURI) ?? compatible)
SanctionedNCName  ::= NCName
SanctionedURI     ::= URI
SanctionedName    ::= SanctionedNCName SanctionedURI [description]
```

The solution to this lies in a combination of the CTS2 identification mechanism and the establishment of one or more external registries, all of which implement the function:

F1(Type, Identifier) → SanctionedName

The purpose of F1 is, in the context of type Type, to return the official SanctionedName for *any* known identifier, be it a NCName, a URI or any other recognized string. The data types used in this specification and the ones that follow are described below.

Part of the key to function f1 as defined above is that the maps:

F1(Type, SanctionedNCName) → SanctionedNCName

and

F1(Type, SanctionedURI) → SanctionedURI

are always included part of the function meaning that, as long as f1 is used consistently by a client—that, while it may cache the results, if it is capable of re de-referencing all URI's and identifiers, it becomes possible for the implementing service to update SanctionedNames over time.

Examples of function F1

Arguments	**Result**
F1(CODE_SYSTEM, '2.16.840.1.113883.6.96')	'SNOMED_CT', 'http://snomed.info/sct/900000000000207008', 'SNOMED CT International Edition'
F1(CODE_SYSTEM, 'urn:oid:2.16.840.1.113883.6.96')	'SNOMED_CT' 'http://snomed.info/sct/900000000000207008' 'SNOMED CT International Edition'
F1(CODE_SYSTEM, 'SNOMED Clinical Terms')	'SNOMED_CT' 'http://snomed.info/sct/900000000000207008' 'SNOMED CT International Edition'
F1(CODE_SYSTEM, 'SNOMED_CT')	'SNOMED_CT' 'http://snomed.info/sct/900000000000207008' 'SNOMED CT International Edition'

Arguments	Result
F1(CODE_SYSTEM, 'SNOMEDCT')	'SNOMED_CT' 'http://snomed.info/sct/900000000000207008' 'SNOMED CT International Edition'
F1(CODE_SYSTEM, 'http://umls.nlm.nih.gov/ RSAB/SNOMEDCT')	'SNOMED_CT' 'http://snomed.info/sct/900000000000207008' 'SNOMED CT International Edition'
F1(CODE_SYSTEM, 'http://snomed.info/sct/ 900000000000207008')	'SNOMED_CT' 'http://snomed.info/sct/900000000000207008' 'SNOMED CT International Edition'
F1(CODE_SYSTEM, 'http://purl.bioontology.org/ ontology/SNOMEDCT')	'SNOMED_CT' 'http://snomed.info/sct/900000000000207008' 'SNOMED CT International Edition'
F1(CODE_SYSTEM, 'UMLS')	'UMLS' 'http://umls.nlm.nih.gov/RSAB/MTH' 'UMLS Metathesaurus'
F1(CODE_SYSTEM, 'http://umls.nlm.nih.gov/RSAB/M TH')	'UMLS' 'http://umls.nlm.nih.gov/RSAB/MTH' 'UMLS Metathesaurus'
F1(CODE_SYSTEM, 'MTH')	'UMLS' 'http://umls.nlm.nih.gov/RSAB/MTH' 'UMLS Metathesaurus'
F1(CODE_SYSTEM_VERSION 'SNOMEDCT_2013_01_31')	'SNOMED_CT_20130131' 'http://snomed.info/sct/900000000000207008/version/2013013 1' 'SNOMED CT International Edition January 2013 Re- lease'
F1(SOURCE 'Mayo Clinic')	'Mayo_Clinic' 'http://www.mayoclinic.org' 'Mayo Clinic'
F1(SOURCE '2.16.840.1.113883.3.2')	'Mayo_Clinic' 'http://www.mayoclinic.org' 'Mayo Clinic'
F1(NAMESPACE 'SNOMED_CT')	'SCTID', 'http://snomed.info/id/'
F1(NAMESPACE 'SCTID')	'sctid', 'http://snomed.info/id/'
F1(NAMESPACE 'http://snomed.info/id/')	'sctid', 'http://snomed.info/id/'
F1(NAMESPACE 'OWL')	'owl' 'http://www.w3.org/2002/07/owl#'

4.2 Sanctioned Name to Known Identifiers

The second function, F2 is the inverse of the above - given a sanctioned name, it returns all of the known identifiers and their sources:

F2(Type, SanctionedNCName) → (Identifier (Format, Source)*)

Examples:

Arguments	Result
F2(CODE_SYSTEM, 'SNOMED_CT')	'2.16.840.1.113883.6.96', (OID, HL7), 'urn:oid:'2.16.840.1.113883.6.96', (OIDURI, HL7), 'SNOMED Clinical Terms', (LABEL, HL7), 'SNOMED_CT', (LABEL, IHTSDO), 'SNOMEDCT', (RSAB, NLM), 'SNOMEDCT', (LABEL, NCBO), 'http://umls.nlm.nih.gov/RSAB/SNOMEDCT', (URI, NLM), 'http://snomed.info/sct/900000000000207008', (URI, IHTSDO), 'http://purl.bioontology.org/ontology/SNOMEDCT', (URI, NCBO)

4.3 Resource and Version Identifier to Sanctioned Name

The third function, F3 takes the SanctionedNCName of a versionable resource (e.g. CODE_SYSTEM, MAP, VALUE_SET) and an Identifier that represents a version of that re-source and returns the SanctionedName of the resource version.

F3(Type, SanctionedNCName, Identifier) → SanctionedName

Examples:

Arguments	Result
F3(CODE_SYSTEM, 'SNOMED_CT', '20130131')	'SNOMED_CT_20130131' 'http://snomed.info/sct/900000000000207008/version/20130131' 'SNOMED CT International Edition January 2013 Release'
F3(CODE_SYSTEM, 'UMLS', '2012AA')	'MTH2012AA' 'http://umls.nlm.nih.gov/RSAB/MTH2012AA' 'UMLS Metathesaurus 2012 Release 1'

4.4 Retrieving Known Version Identifiers

The next bit of knowledge that is needed is, given a code system, map or value set, to return the sanctioned names of all of the known versions of the resource. Note that this isn't the same as the CTS2 function calls, as the primary role of this service is to provide "sanctioned" identifiers for various resources. Note also, that once the SanctionedNCName is known for a given resource, the F24.2 function can be used to discover all of the identifiers it is known by.

Function F4 takes a Type and the sanctioned name of a versionable resource and returns a list of the Name/URI/description tuples of all known versions of that resource:

F4(Type, SanctionedNCName) → SanctionedName*

Example The example below returns all of the known versions of the Unified Medical Language System ??

Arguments	Result
F4(CODE_SYSTEM, 'UMLS')	('MTH2012AB', 'http://umls.nlm.nih.gov/RSAB/MTH2012AB', 'UMLS Metathesaurus 2012 Release 2'), ('MTH2012AA', 'http://umls.nlm.nih.gov/RSAB/MTH2012AA' 'UMLS Metathesaurus 2012 Release 1', ('MTH2011AB', 'http://umls.nlm.nih.gov/RSAB/MTH2011AB' 'UMLS Metathesaurus 2011 Release 2'), ...

5 Concluding Remarks

References

1. http://www.json.org/.
2. Technical report, ISO/IEC.
3. Bfo basic formal ontology. http://www.ifomis.org/bfo.
4. Health level seven international. http://www.hl7.org.
5. Integrated terminology system. http://www.itserver.es/ITServer/.
6. Logical observation identifiers names and codes (loincOR). http://loinc.org.
7. National cancer institute nci term browser. http://ncit.nci.nih.gov/.
8. Ncbo bioportal. http://bioportal.bioontology.org/.
9. Tim Berners-Lee. What do http uris identify? http://www.w3.org/DesignIssues/ HTTP-URI, jun 2002.
10. Tim Berners-Lee. What http uris identify. http://www.w3.org/DesignIssues/ HTTP-URI2, jun 2005.
11. Tim Bray, Richard Tobin, Henry S. Thompson, Dave Hollander, and Andrew Layman. Namespaces in XML 1.0 (third edition). W3C recommendation, W3C, December 2009. http://www.w3.org/TR/2009/REC-xml-names-20091208/.
12. Roberto Chinnici, Sanjiva Weerawarana, Jean-Jacques Moreau, and Arthur Ryman. Web services description language (WSDL) version 2.0 part 1: Core language. W3C recommendation, W3C, June 2007. http://www.w3.org/TR/2007/REC- wsdl20-20070626.
13. Common terminology services 2 (CTS2). http://www.omg.org/spec/CTS2/1.0, 2012.
14. 14. Cts2 core model elements, v1.0. http://www.omg.org/cgi-bin/doc?formal/ 12-11-04, 2012.
15. Common terminology services 2 (CTS2) version 1.1. http://www.omg.org/spec/ CTS2/1.1, 2013. Still in progress - not yet available

16. Uche Ogbuji Eric Miller. Icd uri model for naming and supporting web services. Document v 1.1, june 2012.
17. Roy T. Fielding. Architectural styles and the design of network-based software architectures. PhD thesis, University of California, Irvine, 2000.
18. Roy T. Fielding. http://lists.w3.org/Archives/Public/www- tag/2005Jun/0039.html, jun 2005. httpRange-14 Resolved, e-mail archive.
19. Roy T. Fielding and Richard N. Taylor. Principled design of the modern web architecture. ACM Trans. Internet Techn, 2(2):115–150, 2002.
20. Centers for Disease Control and Prevention (CDC). International classification of diseases, ninth revision, clinical modification (icd-9-cm).
21. Marc Hadley. Web application description language. Technical report, W3C, Au- gust 2009. http://www.w3.org/Submission/2009/SUBM-wadl-20090831/.
22. National Cancer Institute. Lexevs. https://wiki.nci.nih.gov/display/LexEVS/ LexEVS. Web page accessed 4/1/2013.
23. ISO 3166 - Country Codes. http://www.iso.org/iso/country_codes/.
24. Lexicon query service (LQS). http://www.omg.org/spec/LQS/1.0/, 2000.
25. M. Suignar" M. Duerst. Rfc 3987 - internationalized resource identifiers (iris). Technical report, W3C, jan 2005. http://www.ietf.org/rfc/rfc3987.txt.
26. Eric Miller. Icd resource identification proposal v0.1. Correspondence with WHO, may 2011.
27. National Library of Medicine.
28. Jyotishman Pathak, Harold R. Solbrig, James D. Buntrock, Thomas M. Johnson, and Christopher G. Chute. Implementation brief: Lexgrid: A framework for representing, storing, and querying biomedical terminologies from simple to sublime. JAMIA, 16(3):305–315, 2009.
29. A. Phillips and M. Davis. Tags for Identifying Languages, September 2009.
30. L. Masinter R. Fielding. Uniform resource identifier (uri): Generic syntax. Technical report, The Internet Society, jan 2005. http://www.ietf.org/rfc/rfc3986.txt.
31. ED Ray Gates. Iso/iec 11179-3:2013 information technology – metadata registries (mdr) – part 3: Registry metamodel and basic attributes. Technical report, Inter- national Standards Organization, 2013.
32. Leonard Richardson and Sam Ruby. RESTful Web Services. O'Reilly, 2007.
33. Harold Solbrig, Tony Weida, Larry Streepy, David Markwell, and Keith Campbell. Hl7 common terminology services. Technical report, Health Level Seven (HL7OR), 2004.
34. Cui Tao, Jyotishman Pathak, Harold R. Solbrig, Wei-Qi Wei, and Christopher G. Chute. Terminology representation guidelines for biomedical ontologies in the semantic web notations. J. of Biomedical Informatics, 46(1):128–138, February 2013.