An Introduction to Common Terminology Services Release 2 (CTS2)

Craig Stancl

Mayo Clinic
CTS2 Introduction

WHY CTS2?
Challenges in Terminology

• Terminologies vary considerably in content, structure, and purpose

• User requirements of terminology differ (real time decision support, reimbursement, lab, etc…)

• Storage formats differ (relational database, XML, MIF…)

• Even using a standardized terminology (i.e. SNOMED) it is necessary to ensure that terminology content is accessed and distributed consistently

This is where terminology services come in…
Common Terminology Services

• Standardized interface specification for accessing terminologies

• Identify an agreed upon set of functional characteristics for a terminology service

• Separate the terminology content from terminology functionality

• Provide a common entry point for terminology access and maintenance
Common Terminology Service Value

• Enables applications to have access to the same terminology sources

• Ensures that applications access terminology content the same way

• Avoids duplication of
  • Deployed terminologies
  • Software development and deployment (maintenance)

• Provides control over
  • What terminologies can be accessed by applications

• Helps enable terminology interoperability across domains
Evolution

Common Terminology Services - Release 2

Evolved from:

• OMG LQS Specification (1999)
  • OO Model, read only, but laid most of the groundwork

• HL7 CTS Specification (2004)
  • ANSI and ISO Standard
  • SOA Model, read only, reduced scope from LQS
Common Terminology Services Release 2 (CTS2)

A standard for a shared semantic model and API for the query, interchange and update of terminological content.

Terminological content: code sets, value sets, lexicons, thesauri, classification systems, ontologies, …
CTS2 Goals

Specify a common model of what is common amongst these resources

Include metadata about what the resources are for, who publishes them, how often they are released

Create mechanisms for federation, distribution, incremental update and history
CTS2 Introduction

PROCESS
Healthcare Service Specification Project

Developed through the Healthcare Services Specification Project (HSSP) - a collaboration between Health Level 7 (HL7) and the Object Management Group

- HL7 provides the requirements as a Service Functional Model
- OMG develops the formal specification
- HL7 adopts and validates via an HL7 Implementation Guide
Healthcare Services Specification Project (HSSP) Workflow

1. HL7 SFM
2. OMG RFP
3. Vendor Community
   - We Are Here...
4. HL7 Standard
5. Final Standard
6. Corrections / Clarifications

HL7 Implementation Guide

© Copyright 2011, Mayo Clinic
CTS2 Introduction

GUIDE TO THE SPEC
OMG CTS2 Beta Standard

CTS2 is an *Application Programming Interface* (API) specification.

• It defines the semantics, syntax and valid interactions that can occur

• CTS2 is *not* software - it is a “blueprint” for building and using software

• If everyone follows the blueprint (and the blueprint is sufficiently precise) then CTS2 clients and services can interoperate
OMG CTS2 Specification
Resource Oriented Architecture

Based on Resource Oriented Architecture (ROA)

• aka. Representational State Transfer (REST) *Architectural Style*

• An approach to modeling – do not confuse with “anything that is accessed via HTTP URL’s”
OMG CTS2 Specification

ROA Gives Us:

• Modular Implementation – build/use only what you need
  • Resources
  • Functionality
  • Representation

• Distribution

• Simplicity (on a PIM level)
OMG CTS2 Specification
Other Key Points

• Generic – NOT healthcare specific
• Supports Semantic Web – RDF and OWL2
• Not intended to be constraining
  • Extensions are ok – in fact encouraged!
  • Purpose is not to say what *can* be done, but rather to say how common things can be done consistently
CTS2 Scope

- SNOMED-CT
- Gene Ontology
- ICD-10
- HL7
CTS2 Scope
CTS2 Scope

- SNOMED-CT
- HL7
- Gene Ontology
- ICD-10
CTS2 Scope

Generality means that some things aren’t covered
Specification Components

• Platform Independent Model (PIM)
  • Static specification (Information Model)
  • Behavioral specification (Computational Model)

• Platform Specific Models (PSMs)
  • Representation (Schema)
  • Signatures (Interface Specification
    • (WSDL / WADL / …)
Platform Independent Model Static Specification

• UML Class Diagrams – classes, attributes and associations (no methods)

• Textual Description – what each element means, where it came from, etc.

• Value Sets and Bindings – valid (or recommended) codes and URI’s that can be used (via CTS2) to get their meanings

• Invariants – what is always true about a CTS2 resource
Code System Catalog Entry

![UML Model](image)

Figure 1.1: Code System Catalog Entry
A code system is a resource that is maintained by individuals and/or organizations, typically has a specific goal or purpose and is published and/or updated at periodic intervals. Its purpose is to declare a collection of codes or identifiers that represent classes, categories, or individuals that are used for reporting, organizing and/or reasoning about knowledge in some discipline, specialty or domain. The CodeSystemCatalogEntry model carries metadata about the code system itself, while CodeSystemVersionCatalogEntry carries information about the content of a code system.

**Class CodeSystemCatalogEntry**

Metadata and access information about a code system.

**Superclasses:**

- Every instance of CodeSystemCatalogEntry is also an instance of AbstractResourceDescription.

**Attributes:**

- codeSystemName - the local identifier that uniquely identifies the code system within the context of the implementing service. Note that the about URI is the globally unique identifier.
- codeSystemCategory - the category or type of resource that the code system represents.
- ontologyDomain - the subject domain of the ontology
- ontologyType - the nature of the content of the ontology
- designedForOntologyTask - the purpose for which the ontology was originally designed
- hasOntologyLanguage - the formal ontology language
- includes - a reference to an external code system that is included as a component of the referenced code system. The Wine Ontology, as an example, includes the Food Ontology as one of its components. Similarly, the US Edition of SNOMED-CT includes the international edition.
- versions - a DirectoryURI that references the known versions of this code system.
- currentVersion - a reference to the code system version marked as CURRENT in the service instance.
- usedOntologyEngineeringTool - information about a tool used to create the ontology

**Invariants:**

1. The resourceID of a CodeSystem is the codeSystemName.
2. The describedResourceType of a CodeSystem is CODE_SYSTEM.
3. One of the resource types must be owl:Ontology or skos:ConceptScheme.
4. CodeSystems cannot have workflow status.
Platform Independent Model
Behavioral Specification

- UML Method Signatures – inputs and outputs
- Textual Description – what the function does, what the inputs mean
- Preconditions – what must be true before for the function call to apply
- Postconditions – what must be true after the function call occurs
- Exceptions – how precondition failures are reported
Code System Catalog Services

Code System Catalog Read Service

Figure 2.1: Code System Catalog Read

The `CodeSystemCatalogReadService` provides basic access to the CTS2 code system catalog via a locally unique name or a primary or alternative URL.

**Class CodeSystemCatalogReadService**

A service that provides direct read access to the contents of the code system catalog.

**Superclasses:**

- Every instance of `CodeSystemCatalogReadService` is also an instance of `BaseReadService`.

**Invariants:**

1. Service must be present in `supportedProfile` list.
2. Every code system name and uri must be unique at a given point in time.
Operation: exists

Determine whether a catalog entry exists that has an about or alternateId that matches the supplied URI or that has a codeSystemName that matches the supplied name that is (was) applicable in the supplied context.

Input Parameters:

- `codeSystemId` - the local name or URI to check for existence (Type: NameOrURI)
- `context_{OPT}` - the context in which the check is performed. (Type: ReadContext)

Return Type: Boolean

Exceptions:

- `UnsupportedLanguage` - The referenceLanguage is not supported by the service
- `UnknownChangeSet` - The change set specified could either not be read or located by the service.
- `ChangeSetIsNotOpen` - The changeSetContext is recognized by the service, but its state is not OPEN.
- `UnsupportedContext` - One or more changeSetContext is not supported by the service

Preconditions:

1. profile and dates are set to SP_CODE_SYSTEM and the known service dates

Postconditions:

1. The return is true if there was a matching code system at the reference time that met the appropriate activity criteria
Platform Specific Models

• Consist of platform specific mapping of
  • Information Model – example, HTTP REST and SOAP both use XML Schema
  • Computational Model – HTTP REST uses URI rules, SOAP uses interface signatures (WADL)
<!-- CodeSystemVersionCatalogEntry

  <xs:annotation>
    <xs:documentation>
      CodeSystemVersion is a collection of assertions about one or more that if one knows the documentURI assigned to the version and the specific
    </xs:documentation>
  </xs:annotation>
</xs:element>
<xs:complexType name="CodeSystemVersionCatalogEntry">
  <xs:annotation>
    <xs:documentation>
      CodeSystemVersion is a collection of assertions about one or more that if one knows the documentURI assigned to the version and the specific
    </xs:documentation>
  </xs:annotation>
  <xs:complexContent>
    <xs:extension base="core:ResourceVersionDescription">
      <xs:sequence>
        <xs:element name="versionOf" type="core:CodeSystemReference" minOccurs="1">
          <xs:annotation>
            <xs:documentation>
              the name, URI and, if supported by the service, link t
            </xs:documentation>
          </xs:annotation>
        </xs:element>
      </xs:sequence>
    </xs:extension>
  </xs:complexContent>
</xs:complexType>
<resource path="version/{codesystemversionid}">
  <param name="codesystemversionid" type="xsd:string" style="template"/>
  <method name="GET">
    <doc title="CodeSystemVersionCatalogReadService:read">Retrieve the specified
    <request>
      <param name="active" type="xsd:string" style="query">
        <doc title="ReadContext:active">determines whether the query only applie</n>
    </param>
    <param name="changepsetcontext" type="xsd:string" style="query">
      <doc title="ReadContext:changeSetContext">the URI of an open change set 
    </param>
  </method>
</resource>
CTS2 Introduction

CONFORMANCE POINTS
CTS2 Conformance Philosophy

• Implement (or use) exactly what is needed
  • Resources
  • Functionality
  • Representation
CTS2 Resource Profiles

- Code System Catalog Entry
- Code System Version
- Entity Description
- Association
- Map Catalog Entry
- Map Version

- Value Set Catalog Entry
- Value Set Definition
- Concept Domain Catalog
- Concept Domain Binding
- Statement
CTS2 Conformance Points
Behavioral Perspective

- Read – direct access
- Query – search and discovery
- Import/Export – external formats
- Update – incremental update
- History – change history
- Temporal – state of service at point in time
- Maintenance – construct incremental updates
CTS2 Conformance Points
Representational Perspective

• XML
  • XML Schema
  • ISO 21090*

• JSON*
• RDF*
• POJO*

* Not present in Beta 1.0 Specification
CTS2 Introduction

CTS2 DEVELOPMENT FRAMEWORK
CTS2 Development Framework

- Under development by Mayo Clinic
- Allows for rapid creation of CTS2 compliant applications.
- Plug-in based – only implement the functionality that is required.
- Uses Model View Controller (MVC) architectural pattern
CTS2 Development Framework
“View” Component

• Implements the static portion of the CTS2 model
  • CodeSystemCatalogEntry, ...
  • (Indirectly) enforces some invariants
CTS2 Development Framework
“Controller” Component

• Implements the behavioral portion of the CTS2 model
  • Accepts events
  • Validates invariants
  • Enforces preconditions
CTS2 Development Framework
“Model” Component

• Transforms View (CTS2 PIM) structures into state (aka “backing store”)
  • Enforces post-conditions
  • May also enforce some invariants
CTS2 Development Framework

- A MVC architecture that is compliant with the CTS2 API specification

- Can be used to
  - Implement against different back ends (e.g. RDF, SQL, existing terminology structures or API’s)
  - Specify and/or create different import and export maps (IHTSDO, OWL, …)
CTS2 Development Framework

• Can be used to (continued)
  • Implement new views (21090, cRDF, …)
  • Extend the controller with business rules and workflow constraints
CTS2 Development Framework

Resources

• Wiki:

• Detailed presentation today from 4:00-5:00 PM
CTS2 Introduction

USING CTS2 SERVICES
Using CTS2 Services
CTS2 REST Implementation

- Demonstrate capabilities of CTS 2 REST implementation using NCBO BioPortal.

- Examples of:
  - Browsing
  - Searching
  - Form Completion
  - Mapping
  - Incremental Versioning/Updates
CTS 2 REST Implementation
Incremental Versioning/Updates
CTS 2 REST Implementation
Mapping
Using CTS2 Services
Resources

• Wiki:
  • http://informatics.mayo.edu/cts2/framework/ (In Action Section)

• Detailed presentation today from 2:45-3:45 PM
CTS2 Introduction

IMPLEMENTATION GUIDES
What the CTS2 Specification does NOT do

- Specify how CTS2 content will be represented in a backing store
- Specify how various terminology models and formats are imported and exported
- Specify how specific terminology workflow and business rules are realized in a CTS2 service
CTS2 Implementation Guide

• States how content and structure of a terminological resource maps to the CTS2 information model
  • Could be for import/export
  • Could also apply to backing store

• Identifies terminology specific business rules that services must enforce

• Aligns CTS2 w/ organization workflow

• Identifies any extensions to CTS2 specific to the given terminology
CTS2 Implementation Guides

- Owned by appropriate SDO / terminology development organization

- Purposes:
  - Unambiguous transformation between native SDO representation and CTS2 core
  - Consistent representation of any SDO extensions deemed appropriate
  - Specification of SDO specific business rules and workflow model
HL7 CTS2 Implementation Guide

• HL7 Vocabulary Working Group has approved project to develop the HL7 CTS2 Implementation Guide.
  • Target draft document March 2012

• Guide published and owned by HL7.

• Errors, omissions and questions fed to OMG FTF task force.

• Targeted completion Sept 2012
IHTSDO (SNOMED-CT) has formed a group to develop the SNOMED-CT CTS2 Implementation Guide

• Target draft document Mar 2012
RDF/OWL
CTS2 Implementation Guide

• Targeting RDF/OWL implementation guide middle of 2012
CTS2 Introduction

CURRENT STATE AND NEXT STEPS
Current State

CTS2 Specification

• CTS2 PIM / HTTP REST PSM and SOAP PSM voted in as OMG standard

• CTS2 Beta 1 publically available
  • [http://www.omg.org/spec/CTS2/1.0/Beta1/](http://www.omg.org/spec/CTS2/1.0/Beta1/)
Current State
CTS2 Specification

• OMG Finalization Task Force (FTF) formed and in process.
  • Accepting FTF comments until February 1, 2012.
  • Comments will be reviewed and report issued on May 21, 2012.
  • FTF Recommendation due June 29, 2012.
Current State

HL7 Normative CTS2

• HL7 CTS 2 Service Functional Model DSTU expired November 2011.

• HL7 Vocabulary Working Group has approved project to ballot CTS2 SFM as a Normative standard.
  • Target draft ballot May 2012

• Align SFM with OMG PIM to correct any errors identified in the SFM.

• Targeted Normative Ballot Sept 2012
Current State
Resources

• CTS 2 Wiki
  • http://informatics.mayo.edu/cts2

• OMG Project Page
  • http://www.omg.org/techprocess/meetings/schedule/CTS2.html
Next Steps

• Attend session today from 5:20-5:30 PM to learn about…
  • RDF development
  • CTS 2 to RDF Map
  • Additions to the CTS2 Development Framework