# Common Terminology Services Release 2 (CTS2) In Action

**Contents of this Page**


- Videos
  - CTS2 In Action Video
  - CTS2 Over BioPortal Video
- NCBO Implementation
  - Introduction
  - Project Goals
  - Approach
  - Results
  - Architecture
- CTS2 Implementation Guides
  - HL7 CTS2 Implementation Guide
  - IHTSDO CTS2 Implementation Guide
- CTS2 Prototypes
  - Bioportal Widgets
  - Mapping Browser
  - Search Tool


The sections on this page provide an overview of the current implementations that demonstrate CTS2 REST examples. These examples demonstrate browsing, searching, forms completion, mapping and incremental versioning and updates. We also have a video presentation of these implementations.

## Videos

### CTS2 In Action Video

**Error rendering macro 'multimedia'**

com.atlassian.renderer.v2.macro.MacroException: Cannot find attachment 'CTS2 In Action.swf'

Kevin Peterson
February 2012
Demonstration of various applications using CTS2 services.

### CTS2 Over BioPortal Video

**Error rendering macro 'multimedia'**

com.atlassian.renderer.v2.macro.MacroException: Cannot find attachment 'CTS2 Over BioPortal.swf'

Kevin Peterson
January 2012
Demonstration of a CTS2 compliant wrapper built over NCBO's BioPortal.

## NCBO Implementation

### Introduction

This page contains a summary of the current state of the CTS2/BioPortal wrapper. It covers the goals of the project, the approach and methodology that was used and finishes with a summary of the current state of the project, a discussion of some of the issues that were encountered and a list of what remains to be resolved.

### Project Goals

The NCBO BioPortal was created "to access and share ontologies that are actively used in biomedical communities." To meet this goal, BioPortal has developed the BioPortal REST API , which can be used to access BioPortal using http. One of the primary applications of this API are web browsers that can use Ajax widgets to browse and access ontology content for a variety of uses. Samples of some of these widgets can be found here.

The Common Terminology Services 2 (CTS2) specification was created in response to a set of requirements published by Health Level Seven (HL7) and an RFP that was issued by the Object Management Group. This Platform Independent Model (PIM) was designed to be fully compatible with Fielding's notion of the RESTful Architectural Style and one of the key Platform Specific Models (PSMs) is based on http/REST. The model, documentation, schema and WADL can be found on the home page of this guide.

The NCBO community believes that it will be advantageous to be able to access the BioPortal content through both the existing BioPortal REST API and, where appropriate, the nascent CTS2 REST API. The BioPortal API was used as an one of the inputs to the CTS2 specification. The CTS2 specification was heavily influenced by the LexGrid terminology model and the LexEVS service specification and LexEVS is one of the back end components of the BioPortal implementation. There were, however, decisions made in the CTS2 specification that weren't fully compatible with the existing BioPortal model.

The purpose of this project was to create a mapping between the existing BioPortal API and the corresponding components of the CTS2 REST specification to determine where potential issues and incompatibilities may lie and to use the results of this evaluation to determine (a) the best approach would be to creating a complete, robust CTS2 REST wrapper (b) uncover errors and omissions in the CTS2 specification and (c) to come up with recommendations about how the BioPortal REST API might be enhanced or improved.

## Approach

We began by gathering a list of the key BioPortal resources - Ontology, AbstractConcept, Class, Property and Instance along with various lists. Lacking a formal XML Schema for these resources, we used a combination of sample content from the REST service and the java bean classes for each of the resources to assemble lists of the properties for each of these resources, their types and, where it could be determined their cardinality. We went through these lists, gathering sample input from the REST API - both in form of lists of elements and individual elements. We soon discovered that the content of a resource that appeared in list of resources (e.g. ontologies and an individual resource instance bpr:ontologies/39002 were not the same. Some attributes appeared in both places, some just in the resource instance and some did not appear at all.

We took these lists and created a first shot at the CTS2 equivalents - the results of which can be found in Bioportal ontologies on the Mayo informatics service for the ontology resource and BioPortal terms on the Mayo informatics service for the "terms" (aka. Concept). There were a number of conceptual issues that were uncovered in this process, including:

- CTS2 has a notions of *Code System* and *Code System Version*. While BioPortal has similar concepts - "virtual ontology" and "ontology" in the ontology interface and "ontology" and "ontology version" in the search interface, the "virtual ontology" has no attributes besides its identifier.
- BioPortal treats both full ontologies and subsets derived from full ontologies as instances of "ontology". Lists and queries apply to both types of resource - - a list of the latest version of ontologies on bioontology.org returns both the ontologies themselves as well as all subsets. Similarly, term queries return both the ontology in which the term is defined and any subsets that include that term. CTS2 treats Code Systems (ontologies) and Value Sets (subsets) as separate resources. Lists and queries go against one or the other resource but not both.
- BioPortal assumes that all terms are instances of exactly one of "class", "property" or "instance". CTS2 allows *entity* (the equivalent of "term") to exist without making this distinction. In addition, while the CTS2 REST model does not clearly show how this could be done, the intent of the CTS2 model is to allow an entity to simultaneously be a Class and Instance, Class and Property, etc.

We then created a map from each of the BioPortal REST signatures and the equivalent CTS2 REST signature(s) , wherever possible.

These documents were then used to construct a CTS2 REST Server that used the BioPortal REST services as the back end implementation. In addition, we took a number of the interesting BioPortal Ajax widgets and modified them to use the CTS2 REST api instead. A synopsis of the REST services that were implemented can be found on the BioPortal Wrapper Summary page and a list of the translated Ajax Widgets can be found on the BioPortal Ajax page .

## Results

As expected, a number of issues were encountered in this process including:

- Resource/Resource Version mismatch - discussed earlier
- MetaOntology and MetaOntology mapping - BioPortal has several enumerations (Category, Group, Status) that aren't first class ontologies and, even if they were, might be better served were they drawn from OMV or a similar resource
- URI's - BioPortal has its own URI's but many of these ontologies have one or more "official" URI's drawn from outside sources.
- REST hyperlinks - one of the key aspects of the REST architectural style is to provide the ability to navigate the web of resources without having to know how to construct URI's. As an example, an entity reference in CTS2 carries in it a link to both the code system and the code system version in which it is described. Similarly, the descriptionType, language and any other attribute that references an ontology component has the potential for containing a hyperlink. Constructing some of these hyperlinks from the BioPortal REST service can be non-trivial.

The proposed next steps would be to review the mapping and determine whether to: (a) Complete the remaining tasks using the current wrapper paradigm (b) Re-implement the interfaces against the lower level BioPortal interfaces and databases (c) Produce a hybrid for the time being and focus on an RDF based implementation.

## Architecture

The Bioportal to CTS2 Wrapper is broken into five Core components and one Bioportal-specific implementation component.

The components are:

**Webapp** -- A Spring 3 MVC REST Binding.

Webapp responsibilities include:

- Accepting and routing HTTP requests
- Marshaling/unmarshalling content
- Defining REST signatures (url paths)
- Interpreting error messages into HTTP codes

There are two views available: XML and JSON/JSONP. With JSONP, any JSON request with a 'callback' query parameter will be automatically wrapped in a callback. This is to avoid cross-site-scripting problems as well as integrate into javascript libraries such as YUI and JQuery. As stated, a Spring 3 MVC framework was used as the REST binding framework.

**Filter** -- A generic CTS2 Directory (result) filtering framework. This is to allow different filtering mechanisms related to CTS2 Filters and Restrictions to be generalized. It also supplies various match algorithms.

**Service** -- An API definition that allows an abstract data source (such as Bioportal) to plug into the CTS2 REST framework.

**Util** -- Generic helper classes/constants common to all components

**Rest-Model** -- The CTS2 XML Schema represented by Java Beans built by Castor.

**Bioportal-service** -- The bioportal-specific implemtation of the 'service' API.

The bioportal-service flow is as follows:

1. Accept a service request from the 'webapp'
2. Translates the request into a Bioportal REST call
3. Transform the result of the Bioportal REST call into CTS2 objects
4. Send the CTS2 objects back to the 'webapp' for marshaling and return to client.

The Bioportal REST xml is traversed via the DOM API and converted to CTS2 objects. XPath for Java was also tried, but direct DOM traversals proved much faster. Transform occurs in the edu.mayo.cts2.rest.service.bioportal.transform package. Bioportal calls are cached as needed. For example, any XML that is needed on a regular basis will be persistently cached (such as current versions of ontologyId's, etc). All other Bioportal calls are cached in a Least-Recently-Used basis. For example, a term search Bioportal REST call may be cached for future use, but will be evicted from the cache when the cache reaches a certain size. This is an attempt to limit calls to Bioportal for data that is not expected to change often. Changes to Bioportal are listened for via Bioportal's RSS feed, and caches are evicted accordingly when a change is noticed. There are no persisted artifacts other than a cache file, and these will be stored in the $USER_HOME/.cts2 directory.

# CTS2 Implementation Guides

*Why do we need implementation guides?*

To address 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

*What does an implementation guide provide?*

- 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

## HL7 CTS2 Implementation Guide

As part of the CTS2 development effort in the OMG community, it is necessary to ensure that HL7 content can be accurately represented in CTS2.

To be successful, the following need to be addressed:

- Unambiguous transformation between HL7 representation and CTS2 core.
- Consistent representation of any HL7 extensions deemed appropriate.
- Specification of specific HL7 terminology business rules and workflow that need to be provided by HL7 CTS2 service providers.

This will ensure that HL7 content, even though exposed through different implementations of the CTS2 specification, will be consistent and interoperable.

The goal of the implementation guide is to:

- Specify how the HL7 MIF syntax and semantics map to the CTS2 Model, optionally identifying any HL7 specific extensions
- Specify how the HL7 terminology related business rules and workflow would be represented in CTS2 Service Implementations.

Target Dates:

| Task | Target Date |
|---|---|
| Project Team Inception | 2011 December |
| Complete Draft Implementation Guide | 2012 March |
| Submit for DSTU Ballot | 2012 May Ballot |

| | |
|---|---|
| Consider Comments from the DSTU Ballot | 2012 June |
| Submit to TSC for DSTU Approval | 2012 July |
| Close Project (project end date) | 2012 September WGM |

Project Wiki and Documentation can be found at HL7 Wiki - CTS2 Implementation Guide Project .

### IHTSDO CTS2 Implementation Guide

IHTSDO (SNOMED-CT) has formed a group to develop the SNOMED-CT CTS2 Implementation Guide. A draft document is planned for March 2012.

More information will be posted as it becomes available.

## CTS2 Prototypes

### Bioportal Widgets

Various NCBO Bioportal Widgets have be adapted for use with a CTS2 compliant service, available on the Mayo informatics service .

Included are:

- A form auto-complete field with a link to the Bioportal rendering
- Three form auto-complete fields consisting of:
  - URI resolution
  - Term ID resolution
  - Preferred Name resolution

### Mapping Browser

A terminology mapping browser that allows users to search and browse mappings from SNOMEDCT to ICD10

Features include:

- Selectable match algorithms ('startsWith' and 'contains')
- Search based on source or target ontology
- Search based on description text or code

### Search Tool

A tool to search for text-based matches of vocabulary content

Features include:

- Autocomplete ontology selection
- Search based on a single terminology or all terminologies
- Search results shown in summary and detailed view
- Logging console panel to show actual REST queries being sent to the CTS2 service