



Apelon, Inc.
Suite 202, 100 Danbury Road
Ridgefield, CT 06877

Phone: (203) 431-2530
Fax: (203) 431-2523
www.apelon.com

Apelon Distributed Terminology System (DTS)

Apelon DTS Plug-in for Protégé

Version 4.0
June 16, 2005

Table of Contents

1. Overview	3
2. System Requirements.....	3
3. Terms used in this document	3
4. Changes	4
5. Setup and Installation	6
6. Configuring the DTS Tab Widget	6
7. Connecting to an Apelon DTS Server.....	8
8. The DTS Tab Widget	9
Concept Search (Tab in Left Panel).....	9
Concept Tree (Tab in Left Panel).....	10
Concept Walker (Top Right Panel)	11
Concept View (Bottom Right Panel)	11
9. DTS Slot Widgets	13
DtsConceptSlotWidget.....	13
DtsConceptListSlotWidget	17
DtsCodedValueSlotWidget	18
DtsValueTypeWidget	19
10. Mapped CodedValue Slot Data	22
11. CodedValue “Garbage Collection”	22
Incremental garbage collection	22
Project wide CodedValue garbage collection.....	23
12. CodedValue Mapping Utility	23
Running the utility	23
Optional verification and selection of found/unfound matches	25
13. Configuration options	26
14. Troubleshooting / FAQ.....	27

1. Overview

Apelon's Distributed Terminology System (DTS) provides comprehensive terminology services in distributed application environments. The Apelon DTS Plug-in for Protégé offers Protégé users convenient access to an Apelon Knowledgebase composed of several standard medical terminologies, along with local enhancements if desired. It facilitates the creation and maintenance of mappings from Protégé classes (or instances) to corresponding standard medical concepts. When Protégé is used to author clinical guidelines, this facility promotes understanding, communication and standardization of the guidelines themselves by tying them directly to already standard terms.

2. System Requirements

The Apelon DTS Plug-in has been developed and tested using the following:

- Protégé Version 3.0
- Apelon DTS 3.3
- Java Version 1.4.2
- Access to the Internet over port 2323 on TCP if you are behind a firewall.

3. Terms used in this document

The following abbreviations/phrases/terms will be used in this document:

CodedValue (CV) – A metaclass defined in the HL7DataTypes project that is used to represent a “mapping” to an Apelon DTS Concept

DTS mapped CodedValue – A direct instance of CodedValue that contains the necessary information to find and retrieve a corresponding DTS Concept. The “necessary information” can be found in the [Mapped CodedValue slot data](#) section

Mapped Branch – The Class branch of a Protégé knowledgebase where the Apelon plug-in creates and maintains CodedValue direct instances that map to Apelon DTS Concepts. This branch is created as flat lists of DTS concepts grouped under classes representing each DTS namespace such as LOINC, NDF-RT, SNOMED CT, etc., with hierarchical relationships that mirror those in the Apelon DTS Knowledgebase. The root Class to use for the mapped branch is specified in DTSPugin.xml and described in the [Configuration options](#) section.

Deprecated Branch – The Class branch of a Protégé knowledgebase where the Apelon plug-in moves DTS mapped CodedValues that were once used, but are now unreferenced. This branch is created as a flat list; no hierarchical information is kept here. The root Class to use for the deprecated branch is specified in DTSPugin.xml and described in the [Configuration options](#) section.

4. Changes

Version 4.0 (Latest):

- Updated (from DTS 3.1) to be compatible with DTS 3.3 Editor components and to connect to a DTS 3.3 server.
- As a result of updating to DTS 3.3 the plug-in can show an association tree in the Concept Walker panel, such as the Parent Of association in LOINC.
- The Concept Tree and Walker panels can also show the combined hierarchy of an Ontylog namespace with an Ontylog extension namespace. This is part of the Modular Classifier feature of DTS 3.3 that can classify an entire extension namespace of concepts relative to its linked subscription namespace, for example, SNOMED CT. This replaces the classify-one-added-concept-at-a-time capability of the Run-time Classifier prototype which has been removed from the companion Namespace Extensions plug-in for the DTS Editor.
- Tree and Search panels in the DTS plug-in panel are reconfigured as tabs in the left pane, to enhance visibility of the information in those panels.
- Now uses the SNOMED CT Code in Source property instead of the DTS concept ID to retrieve the full 9-digit SNOMED ID, since the latest SNOMED CT DTS subscription namespace had to truncate the rightmost 3 digits of the SNOMED ID to fit in the DTS concept ID.
- Since it's DTS's job to maintain the hierarchy of concepts, the DTS plug-in no longer attempts to maintain a parallel hierarchy in Protégé. All concepts in a namespace are grouped in a flat list directly under namespace concepts under the Concept class in the Protégé class hierarchy.

Version 3.4:

- Now protects the password from being visible when entered in the DTS server logon dialog.
- To distinguish concept expression concepts from other concepts within a concept expression, the former are displayed in bold under the Concept Expression property for such a concept viewed in the Concept View panel of the DTS tab.
- Drag and drop is now supported from the Concept View panel to the Concept Tree, Walker, Search, and the View panel itself. This includes the various concept detail components such as its synonyms, superconcepts, subconcepts, roles, and concepts appearing within a concept expression.
- If a term (including a synonym) rather than a concept is dragged to the Concept View panel, it is now distinguished with a different icon (cyan asterisk instead of a blue ball). A warning message will now appear if a term rather than a concept is showing in the Concept View panel during concept selection.
- In the Concept View panel tree expansion preferences are now remembered. For example, if Properties or Superconcepts details are expanded, these same branches will be expanded when subsequent concepts are dragged to the panel. If the user saves the Protégé project before exiting, then these preferences will be remembered across sessions.

Version 3.3:

- Fixed display of concept expression concepts in Concept View panel in DTS tab to again display the Boolean expression tree under the Concept Expression property, when used with Protégé 2.1 Beta build 230 or later.
- Now renders run-time classification concepts with superconcepts, subconcepts, and roles determined by the classifier, whose results have been stored in the Run-time Classification – Result property.

Version 3.2:

- Recompilation to work with Protégé 2.1 Beta, and Apelon DTS 3.1.
- The Protégé Plug-in directory structure has been implemented. All Apelon DTS Plug-in files are now located in `.../Protégé/plugins/com.apelon.dts_plugin/*`
- When creating the log config file, `DTSPluginLogCfg.xml`, an absolute path is inserted into the config file. If you move protégé after installing the Apelon plug-in, simply delete the log config file, it will get recreated with the correct new absolute path specs.
- The `Protégé.lax` file does not need to be modified with this version.

Version 3.1:

- In the Concept View panel of the Apelon DTS tab widget it's now possible to view the concept expression tree under the Concept Expression property for a concept expression concept that was created using the separate DTS Editor with the Concept Expressions Plug-in.

Post 3.0, before 3.1:

- When creating DTS Mapped CodedValues, the plug-in now stores the DTS Concept ID in the CV “label” slot. The “code” slot is now used for display only purposes, and will be populated with the DTS Concept ID when referring to SNOMED concepts, and the DTS Concept Code when referring to LOINC and all other knowledgebase concepts.
- The CodedValue utility has been updated so that it can be run on existing projects, and will fix up code/label slots on existing DTS mapped CVs. The UI has also been spruced up a bit, and logging during map operations has been enhanced.
- The `DtsConceptSlotWidget` and `DtsConceptListSlotWidget` have been enhanced so that when dealing with an instance slot, the stock Protégé add button on the widget will bring up the appropriate “add instance” dialog.

Version 3.0:

- The plug-in has been upgraded to use a pre-release version of Apelon DTS 3.0
- A secure Apelon DTS 3.0 server is up and out on the Internet, so local installation of the DTS 3.0 database and server is no longer required.
- The plug-in GUI widgets, i.e., search, concept navigation, concept viewers, have been upgraded to use the latest and greatest DTS 3.0 UI widgets.
- The concept selection slot widgets now create HL7 CodedValue instances, and the correct ancestor hierarchy, when selecting DTS Concepts.

- Addition of CodedValue mapping utility that will find CodedValues created manually, and attempt to map them, adding necessary information, to corresponding DTS Concepts.
- DtsValueTypeWidget – Slot widget that allows selection of DTS Concepts as “allowed parents” for a Class value type slot
- Configurable mapped branch root node
- Configurable deprecated branch root node
- CodedValue displayName slot now used for DTS Concept name, name is concatenation of concept name and namespace name for uniqueness, displayName will be used as “Form Browser Key” for CodedValues.
- Garbage Collection of unused, mapped, CodedValues
- DtsConceptSlotWidget, DtsConceptListSlotWidget, and DtsValueTypeWidget have been enhanced to allow selection and view of non-mapped Protégé classes via the stock Class widget behavior in addition to the custom selection and view of Apelon DTS Concepts.

5. Setup and Installation

The Apelon DTS Plug-in can now be installed and run without a full Apelon DTS installation present. After you have Protégé and JRE 1.4.2 or higher installed:

Download the plug-in files

1. In a Web browser, go to <http://support.apelon.com/support/index.asp>.
2. Enter your organization’s username/password provided to you by Apelon Customer Support.
3. Download ProtegeDtsPlugin4.0.x.zip to your local machine. (x designates the latest minor update of the 4.0 plug-in).

If you have a previous version of the plug-in, delete the folder

.../Protege_x.x/plugins/com.apelon.dts_plugin

where x.x is the current version (at least 3.0) of Protégé you have installed.

Install the plug-in files

Unzip, with the “Use folder names” option selected, the ProtegeDtsPlugin4.0.x.zip file into the .../Protege_x.x/plugins/ directory, for example C:\Program Files\Protege_3.0\plugins. The zip contains the following structure:

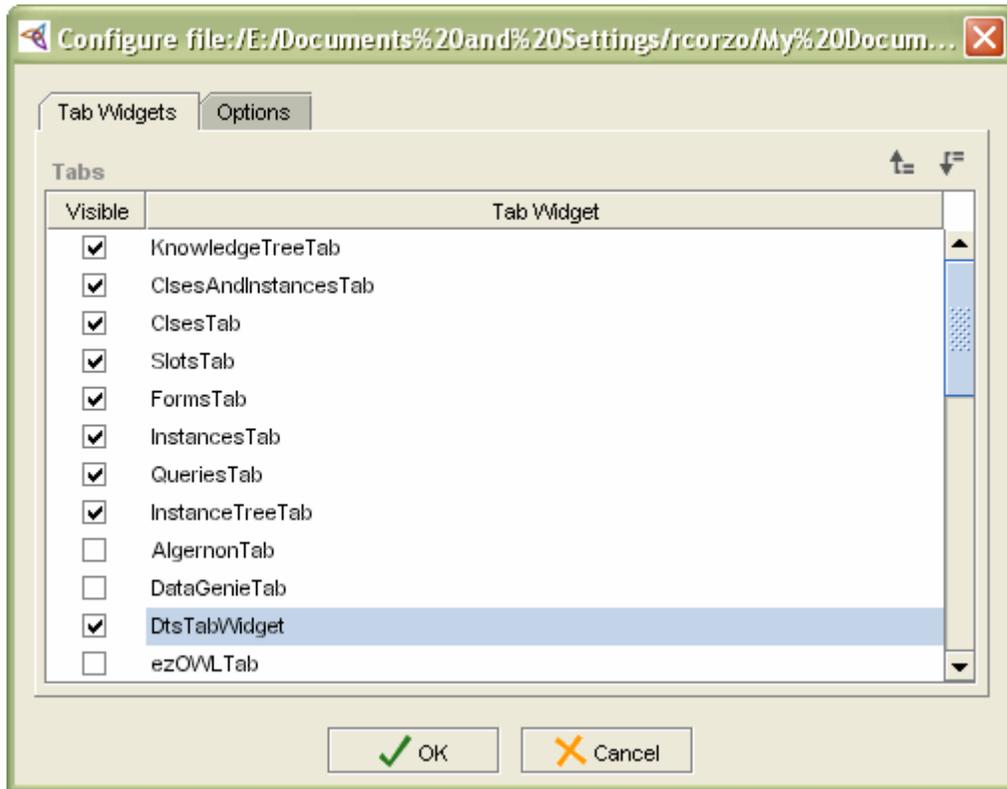
com.apelon.dts_plugin	Plug-in JAR files, config files, and log files
doc	Overview doc and HTML files
DTSPugin_files	Image files for HTML documentation

The com.apelon.dts_plugin/doc directory includes documentation for the DTS Editor, which the plug-in draws from for the UI widgets in the DTS Tab widget. The documentation is written in the context of running the DTS Editor, but there are sections describing the UI widgets that the DTS Plug-in consumes. Specific references to topics in these documents will be made in this document.

6. Configuring the DTS Tab Widget

The Apelon DTS Tab provides the ability to connect to an Apelon DTS Knowledgebase, select namespaces, navigate through concept hierarchies, search for concepts, and inspect concepts. After

you have first installed the Apelon DTS Plug-in, start Protégé, open a project, and then select “Configure...” from the Project menu, which will bring up the project configuration dialog:



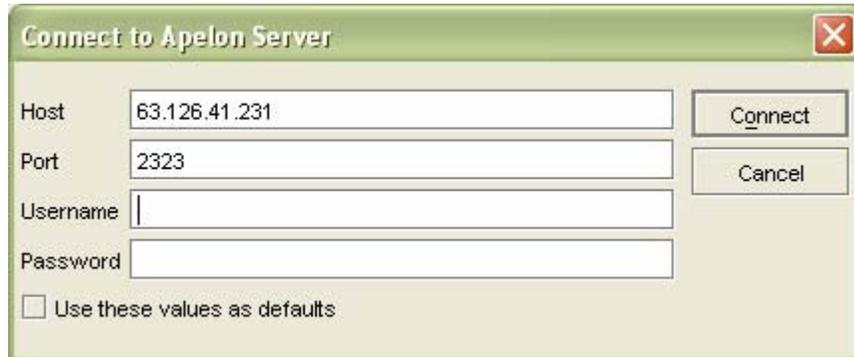
Check the “DtsTabWidget” box, and click OK to close the dialog. This will bring up the Apelon DTS Logon window, so that you can [connect to an Apelon DTS Server](#). (See next section.)

Note: Once your project is configured to load the Apelon DTS tab, you will be prompted to logon each time you open the project.

After logging on, you should see the Apelon DTS Tab.

7. Connecting to an Apelon DTS Server

The Apelon DTS Plug-in connects to an Apelon DTS Knowledgebase via a Secure Socket connection; you will need access to the Internet to connect. When loading the plug-in, it will display the following connection dialog, with the Apelon DTS Server host and port pre-filled.

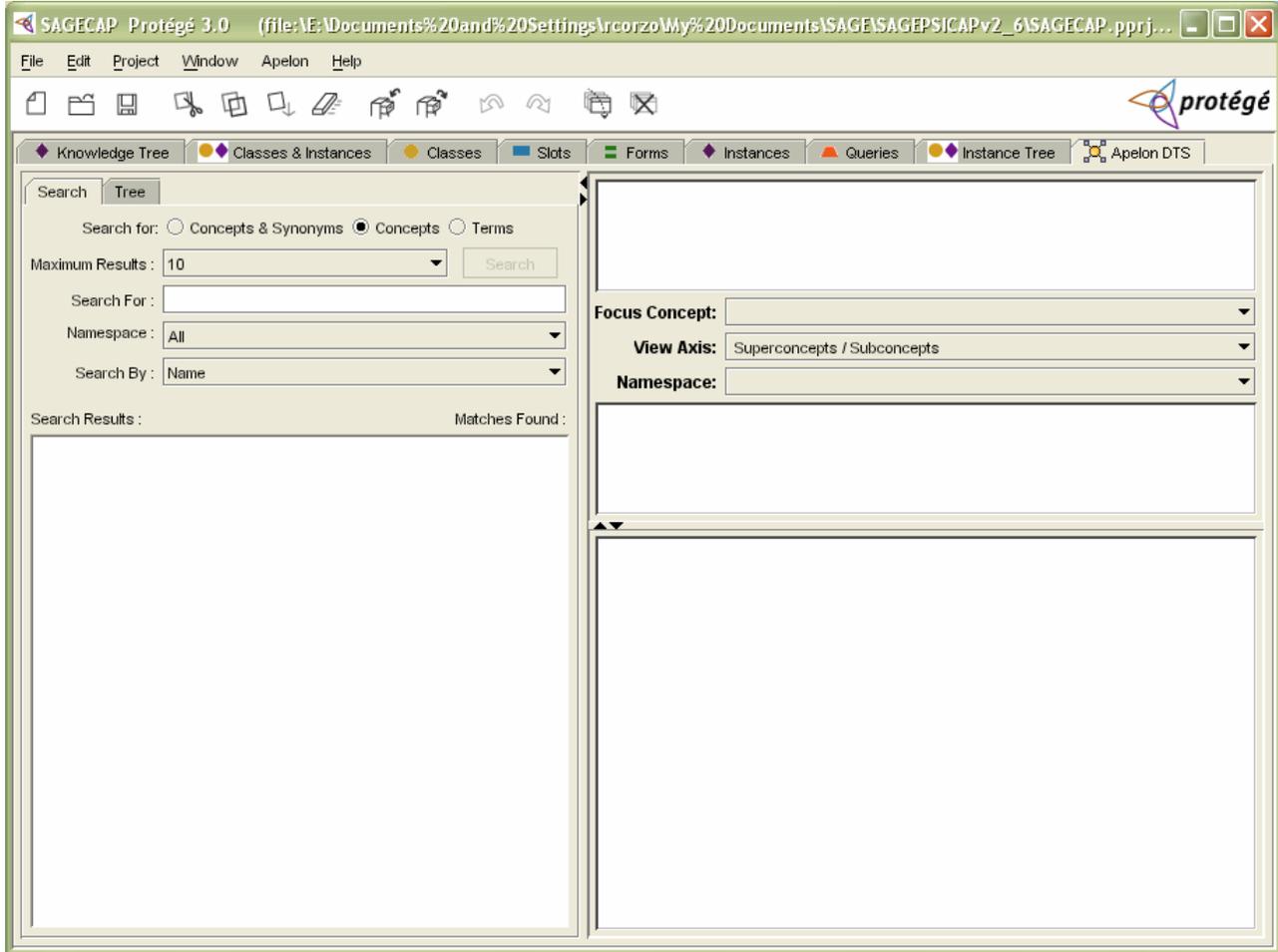


The screenshot shows a dialog box titled "Connect to Apelon Server". It features a standard Windows-style title bar with a close button (an 'X' in a red square). The dialog contains four text input fields: "Host" (pre-filled with "63.126.41.231"), "Port" (pre-filled with "2323"), "Username" (empty), and "Password" (empty). To the right of these fields are two buttons: "Connect" and "Cancel". At the bottom left of the dialog, there is a checkbox labeled "Use these values as defaults" which is currently unchecked.

Enter the username and password by Apelon Customer Support, and then click Connect. (Leave 63.126.41.231 as the host and 2323 as the port.) Ignore the checkbox “Use these values as defaults” as this doesn’t have any function in Protégé.

8. The DTS Tab Widget

After connecting, all tabs selected to be “Visible” should appear. Click on the Apelon DTS tab to bring up the following UI:



As shown above, the Apelon DTS tab comprises four read-only panels laid out as follows:

Concept Search/ Concept Tree	Concept Walker Concept View
---------------------------------	--------------------------------

The panels can be resized by sliding the horizontal and/or vertical divider. The Concept Search and Concept Tree panels are configured as tabs in the left pane. Click on one tab or the other to make that tab visible.

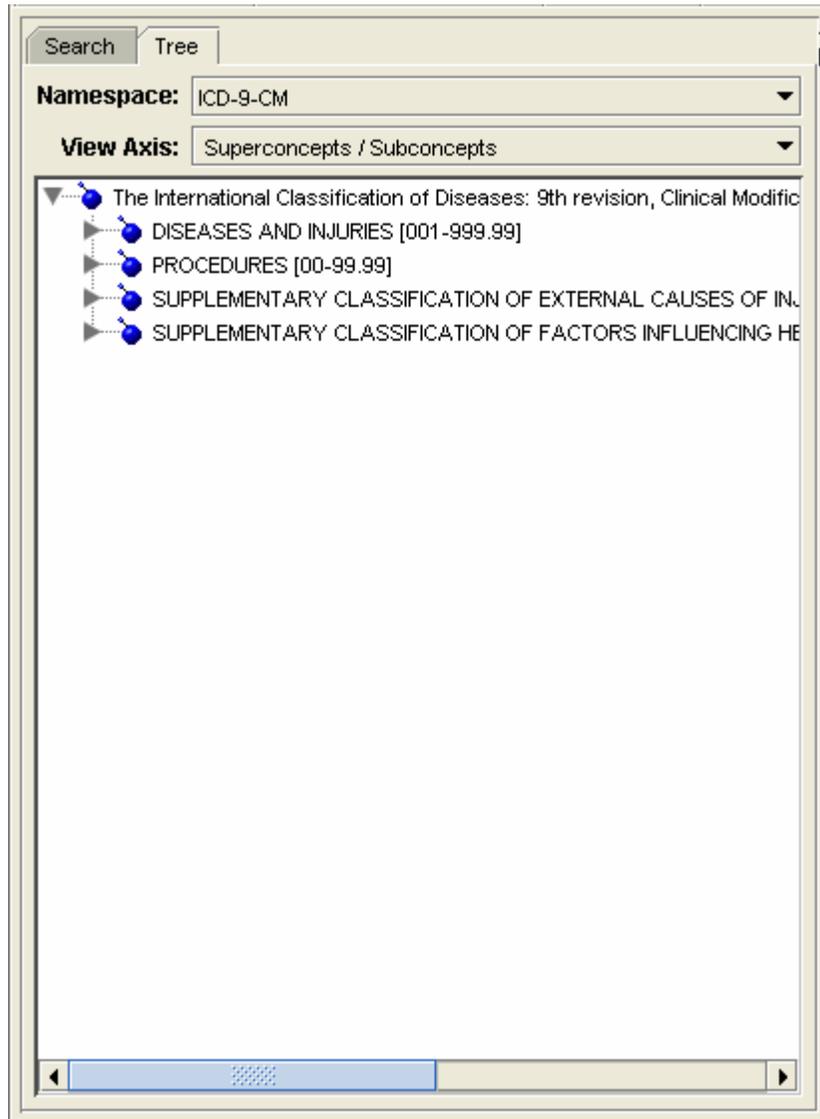
Concept Search (Tab in Left Panel)

The Concept Search panel allows searching for DTS concepts using a variety of search options. The simplest search, search by name allows wildcard searching on DTS Concepts. After searching,

you can drag any concept from the results list box onto the Concept Walker, or the Concept View panel.

See the section “Concept Search” in the plugins/com.apelon.dts_plugin/doc/dtseditor.pdf for specific details.

Concept Tree (Tab in Left Panel)



The Concept Tree panel allows tree navigation of the Apelon DTS Knowledgebase concept hierarchy for the DTS Namespace selected in the Namespace drop down combo box. From this panel, you can select concepts (tree nodes) and drag them onto the Concept Walker or Concept View panel.

See the section “View Concept Tree – *Tree* Tab” in the plugins\com.apelon.dts_plugin\doc\dtseditor.pdf for specific details about the Concept Tree.

Concept Walker (Top Right Panel)

The Concept Walker panel provides a view of the parents and children of a specific DTS Concept. You can drag a concept from the tree panel or the search panel onto the drop down combo in the middle of the Concept Walker panel to navigate a concept. You can drag any tree node in the parent or child tree onto the combo box to set it as the “focus” concept, or you can drag the concept (node) onto the Concept View panel.

See the section “Concept Walker View” in the plugins\com.apelon.dts_plugin\doc\dtseditor.pdf for specific details.

Concept View (Bottom Right Panel)

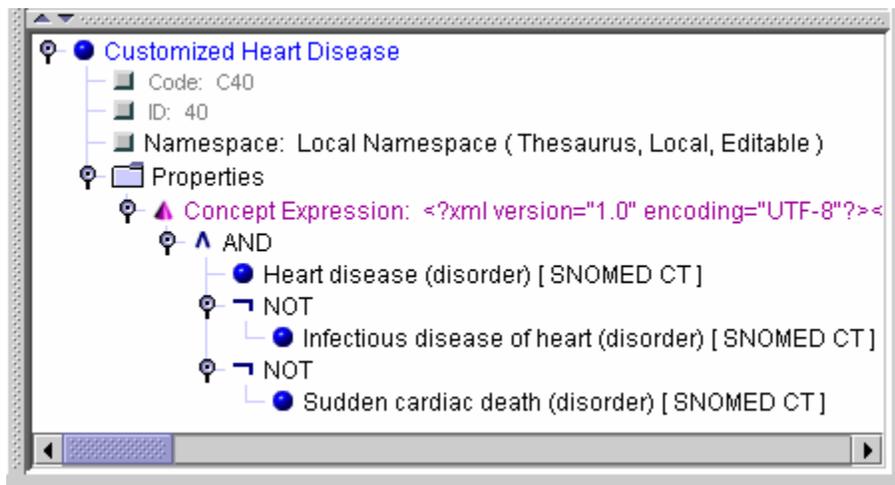
The Concept View panel is a means to inspect that Concept’s Name, Code, and ID, as well as its Role and Property Values, among other things. Concepts can be dragged from the Concept Tree panel, Concept Search panel, and either tree in the Concept Walker panel, and dropped onto the “Concept Name” field of the Concept View panel.

This panel is a read-only version of the Concept/Term Details panel described in plugins\com.apelon.dts_plugin\doc\dtseditor.pdf.

For concepts that were created in the separate Apelon DTS Editor application in a local namespace, as extensions to the content in a subscription namespace, the Concept Viewer will render additional information for these concepts. Examples of such concepts are concept expression concepts and modular classification concepts.

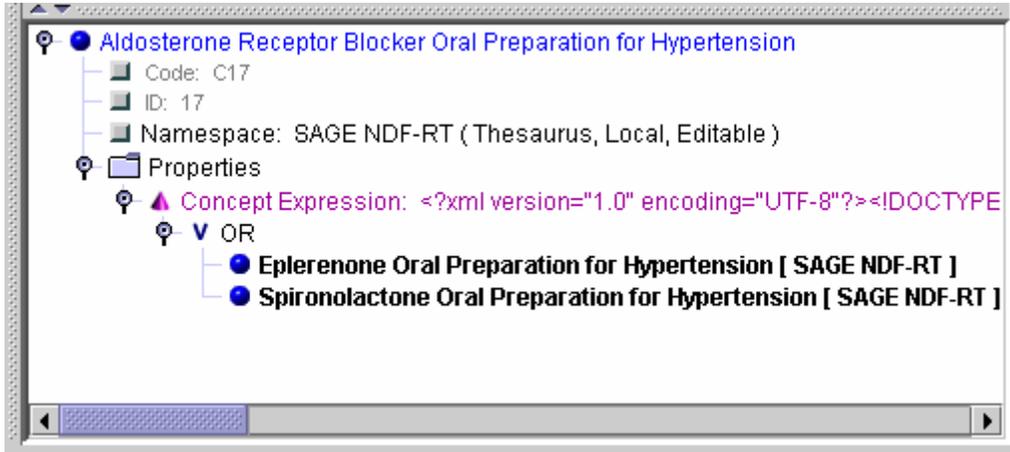
Concept Expression Concepts

If you drag a concept expression concept, which has a value stored in a “Concept Expression” property attached to the concept, you can view the concept expression tree underneath the property in the Concept View by expanding the branches of the concept details tree.



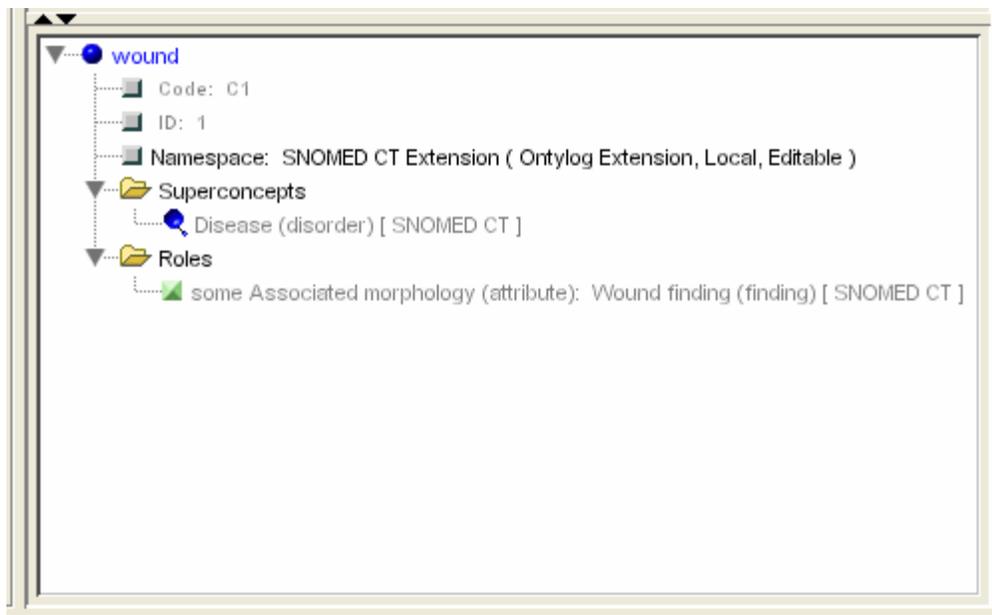
Although “Customized Heart Disease” is defined in “Local Namespace”, the concepts in the Boolean concept expression are drawn from “SNOMED CT”.

If concept expressions concepts are used within a concept expression, they appear in bold:



Modular Classification Concepts

If you drag an Ontylog concept, including one from an Ontylog Extension namespace, you can view its superconcepts, subconcepts, and roles as determined by the Classifier in the Apelon DTS Editor.



Although “wound” is defined in a local namespace such as “SNOMED CT Extension”, the superconcepts, subconcepts, role names and role value concepts may be from a subscription namespace such as SNOMED CT.

Drag and Drop

You can drag any concept (or in some cases a term) from the details in the Concept View to another panel, or even drop it back on the Concept View panel itself. This includes the synonyms, concepts in a concept expression, superconcepts, subconcepts, roles, and associations that appear as details of the Concept View.

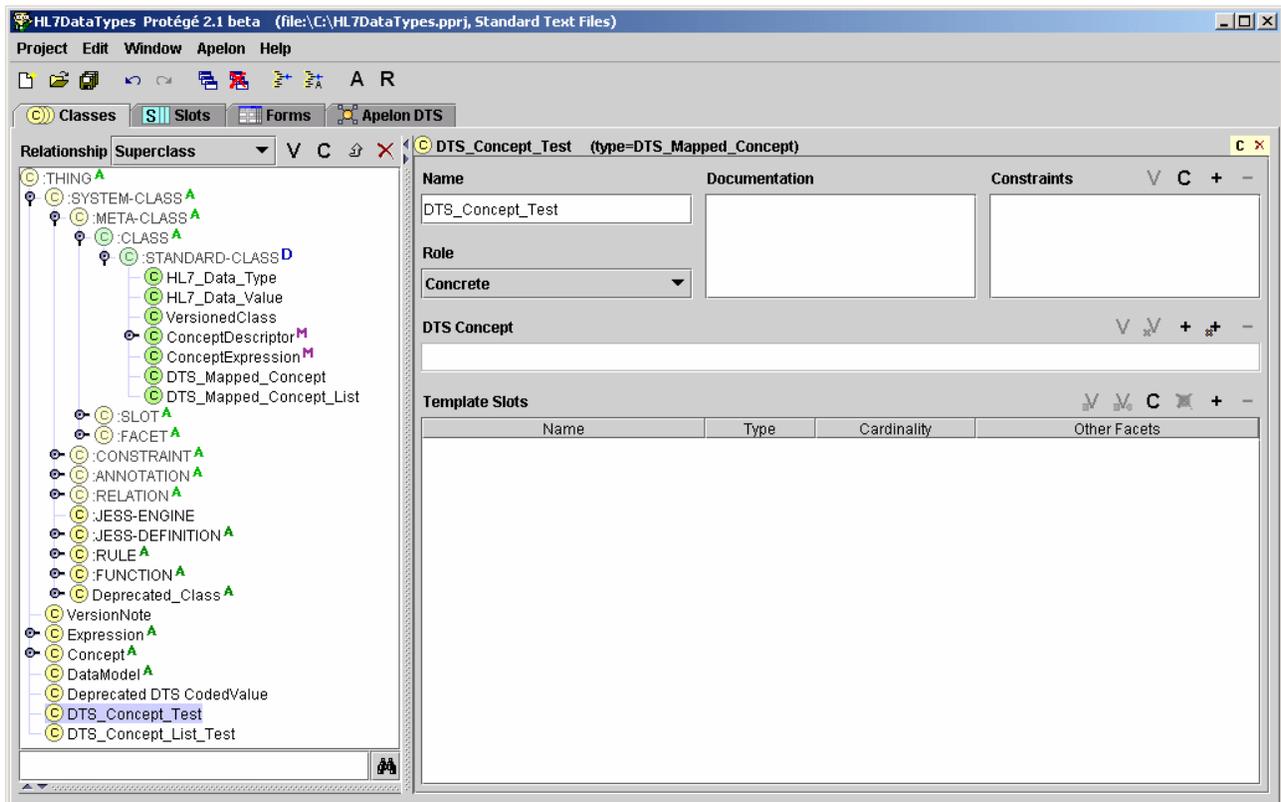
9. DTS Slot Widgets

There are four slot widgets in the DTS Plug-in for referencing Apelon DTS content (concepts) via HL7 CodedValue classes in a Protégé project.

DtsConceptSlotWidget

The DtsConceptSlotWidget is used to create Protégé “CodedValue” classes (the CodedValue metaclass residing in the HL7DataTypes project) that map to DTS Concepts. To accomplish this, in the Protégé Forms tab, you can select any single value class slot, and opt to use the DtsConceptSlotWidget for the slot.

In the following example, we have created a sample “DTS_Mapped_Concept” metaclass, created a Class-valued “DTS Concept” slot for that metaclass, and have configured the slot to use the DtsConceptSlotWidget. We then created a sample class, DTS_Concept_Test, which uses DTS_Mapped_Concept as its metaclass.



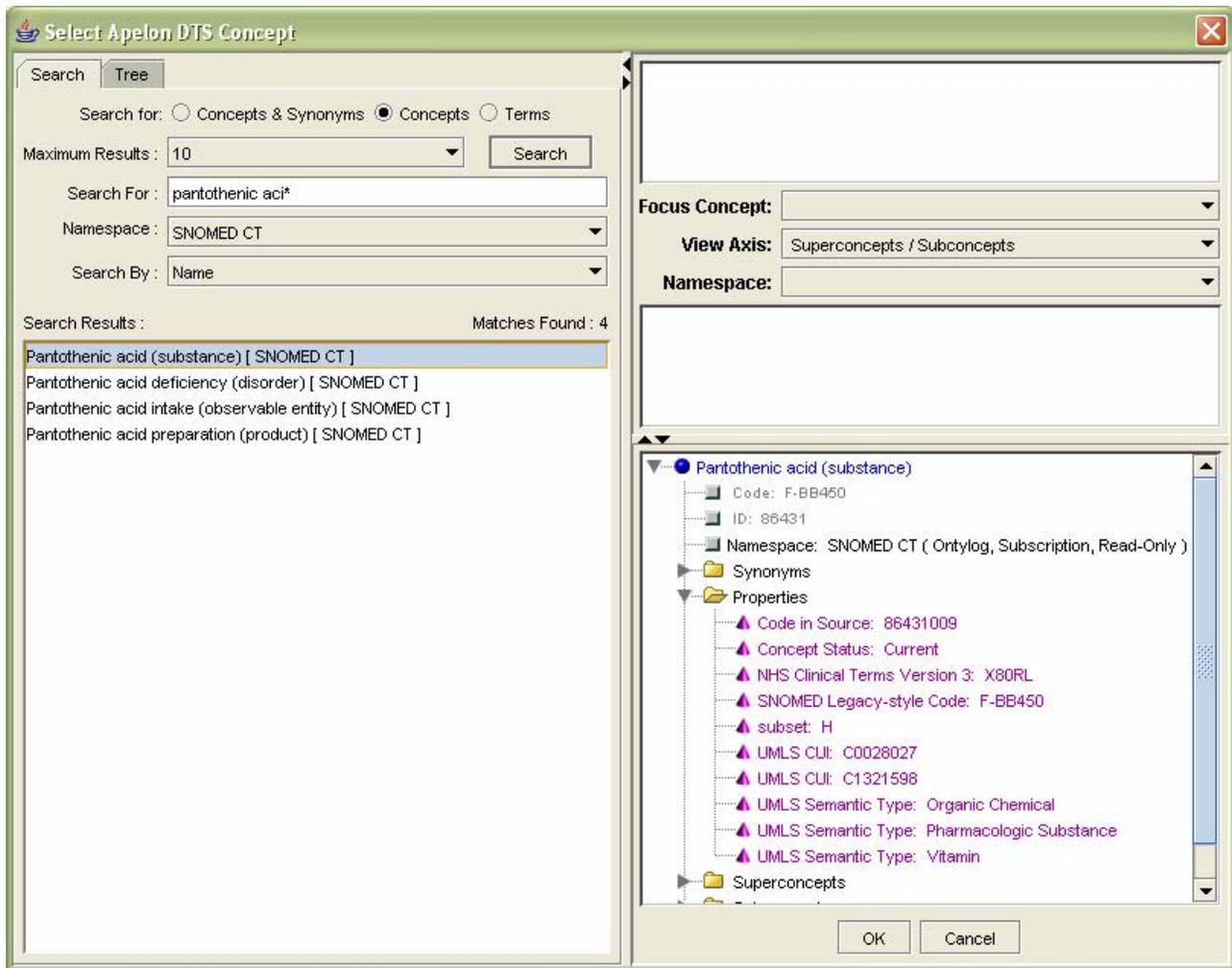
The DtsConceptSlotWidget is an extension of the ClsFieldWidget, performing the same view, add, and remove actions, but with added features for creating and viewing CodedValues that map to Apelon DTS Concept. The buttons of the widget, from left to right, are Cls View, DTS View, Cls Add, DTS Add, and the Cls Remove button. They behave as follows:

Cls View Button - Enabled when the slot has a value assigned. Displays the slot value using the stock ClsFieldWidget behavior.

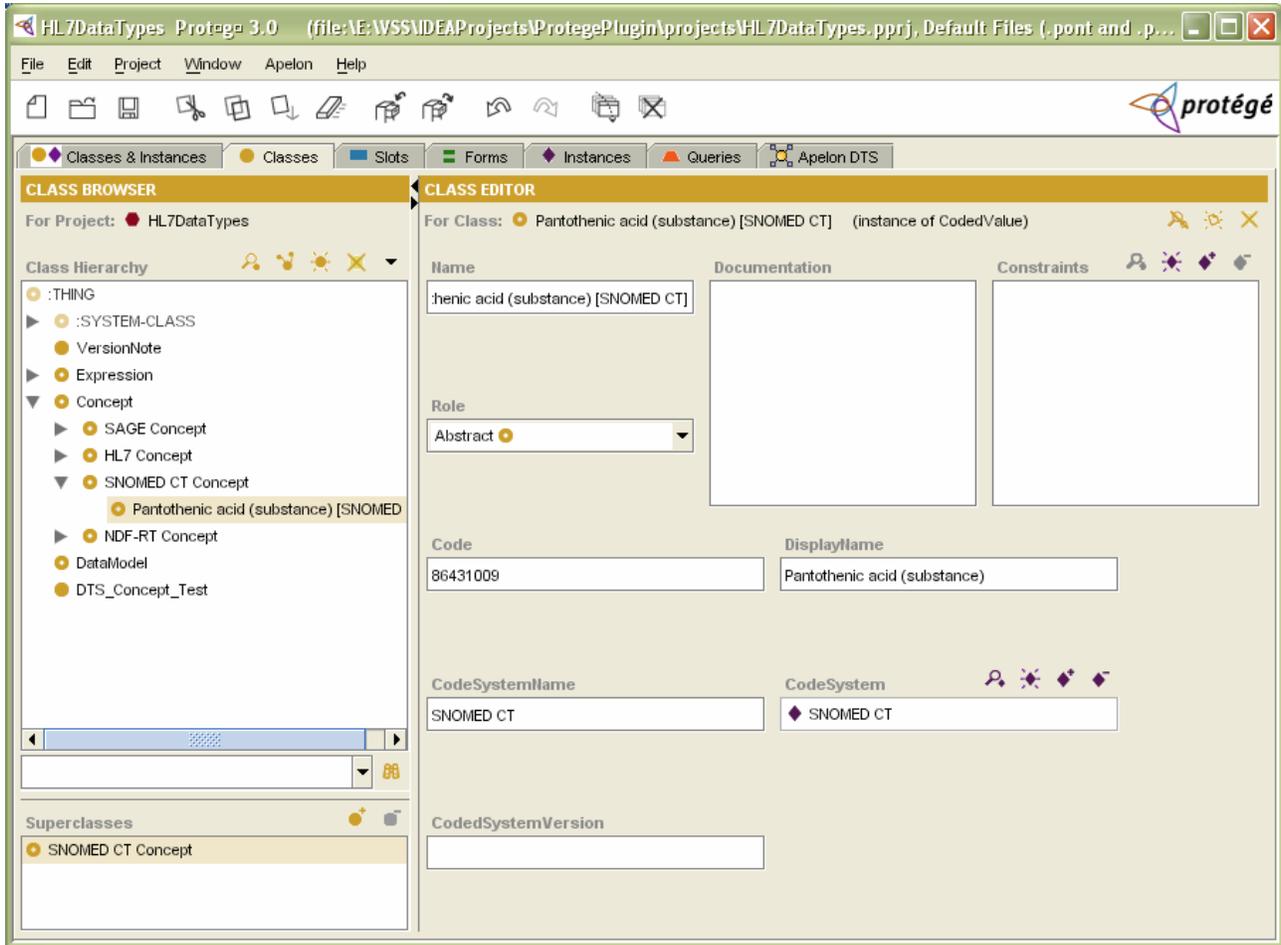
Cls Add Button – Always enabled, except when in the form designer. Selects a slot value using the stock ClsFieldWidget behavior.

Cls Remove Button – Enabled when the slot has a value assigned. Removes the current slot value. If a DTS Mapped CodedValue, Garbage Collection is performed if enabled. See [CodedValue](#) [“Garbage Collection”](#) for details.

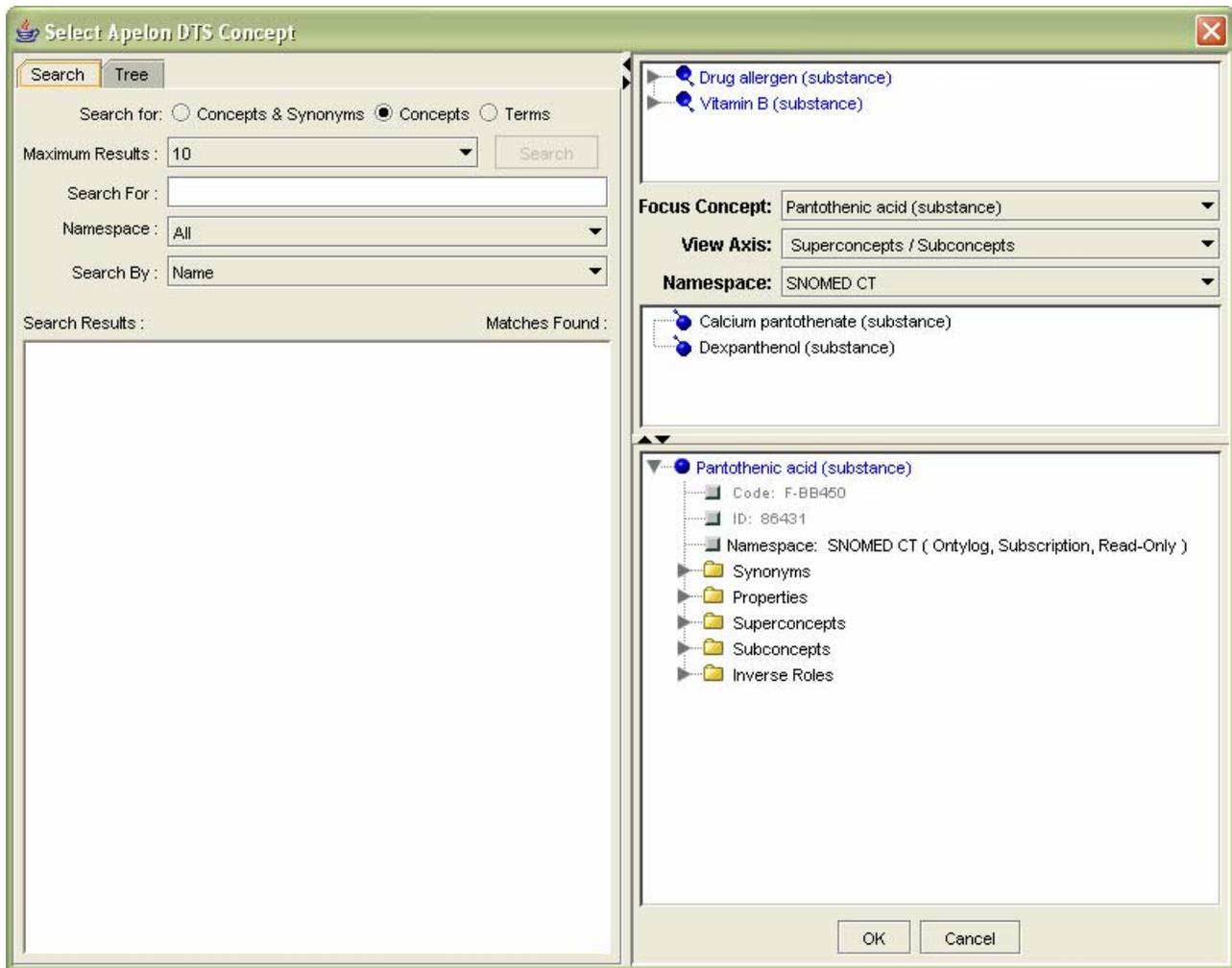
DTS Add Button – Always enabled, except when in the form designer. This button brings up the Apelon dialog allowing selection of a DTS Concept just like the Tab Widget view, but with an OK/Cancel button in the Concept View panel. After identifying the concept desired in the left hand view pane, click the “OK” button. Ensure that the item in the Concept View is actually a DTS concept (with a blue ball icon) rather than a term (with a cyan asterisk icon), or you will get a warning message and be returned to the DTS Concept selection dialog.



After clicking OK, the widget will attempt to find a CodedValue class representing the selected DTS concept in the Protégé knowledgebase. If not found, the widget creates a CodedValue, with slot values as described in [Mapped CodedValue Slot Data](#), representing the selected DTS Concept, and places it under a class representing the concept’s namespace, in turn under a specific top level Protégé class specified via the [configuration options](#) (in this example shown as Concept). Below is an example of what the Classes tree would look like after selecting the above “Pantothenic acid (substance)” concept.



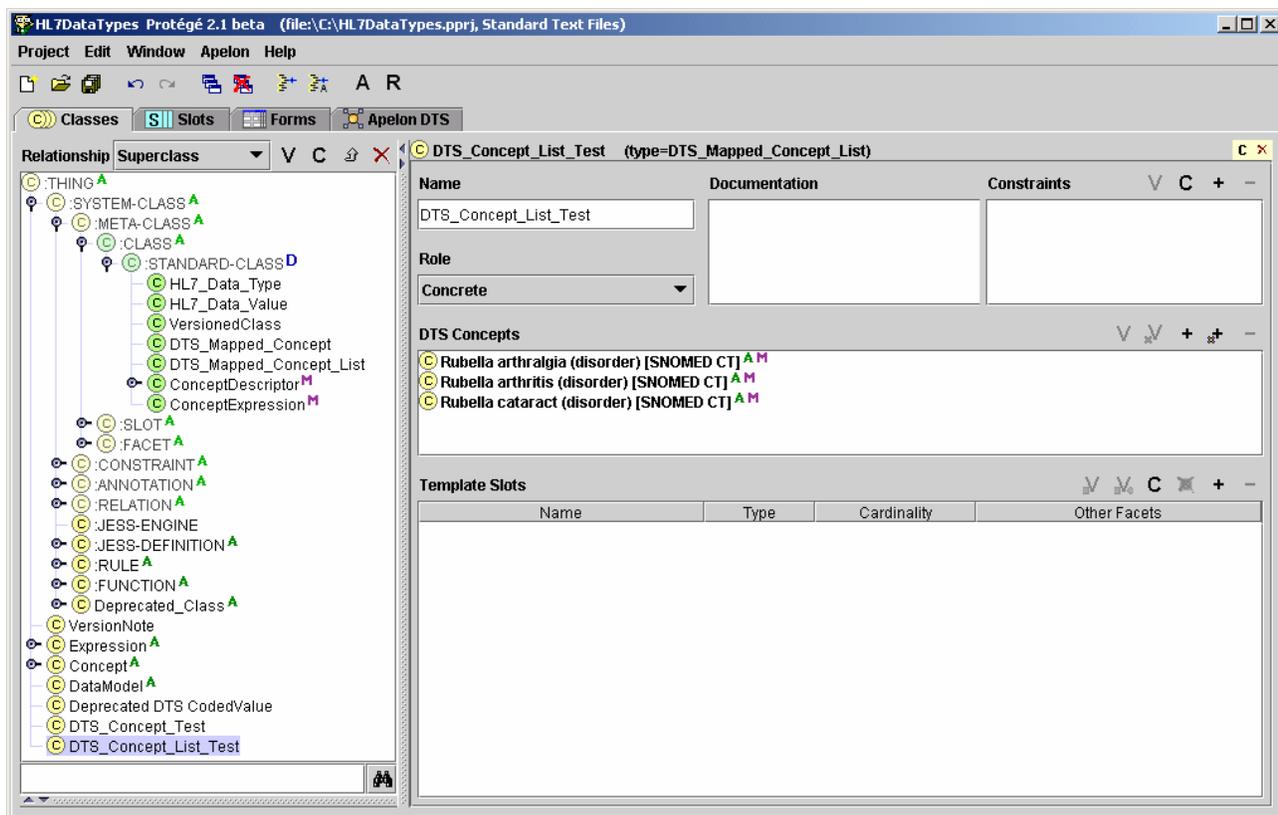
DTS View Button – Enabled when the slot has a value assigned and that value is a DTS Mapped Concept. The DTS View button brings up the same dialog as the DTS Add button, but with the currently selected Apelon DTS Concept in the Concept Walker and Concept View panels. The Select Apelon DTS Concept panel will display the DTS Concept that corresponds to the DTS Mapped CodedValue, displaying that concept in the Concept Walker and Concept View panels as shown below.



You can close the dialog if no changes are necessary, drag a more or less specific concept from the Concept Walker to the Concept View panel, or select an entirely new concept from the Tree View or Search panels. If different than the previous slot value, the widget will replace the old DTS Mapped CodedValue. Garbage collection, if turned on, is called when replacement occurs.

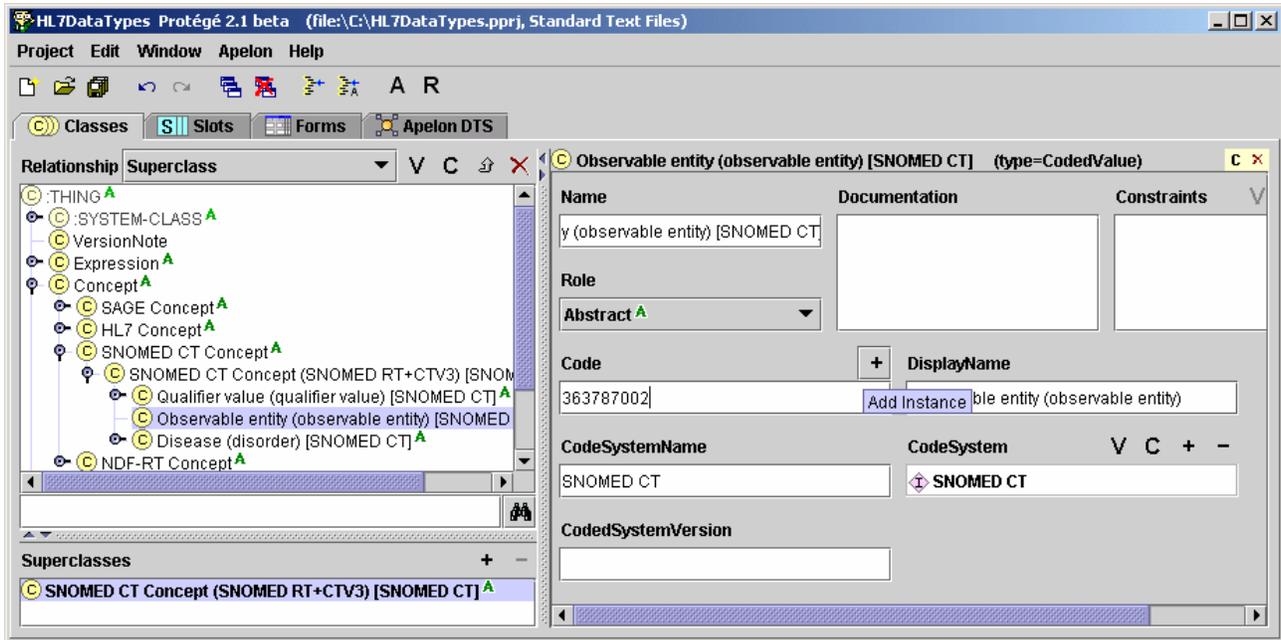
DtsConceptListSlotWidget

This widget is the “list” version of the singular DtsConceptSlotWidget. For the following example, we have created a sample “DTS_Mapped_Concept_List” metaclass, added a “DTS Concept List” slot to that metaclass, and have configured the slot to use the DtsConceptListSlotWidget. We then create a sample class, DTS_Concept_List_Test, that uses DTS_Mapped_Concept_List as its metaclass.



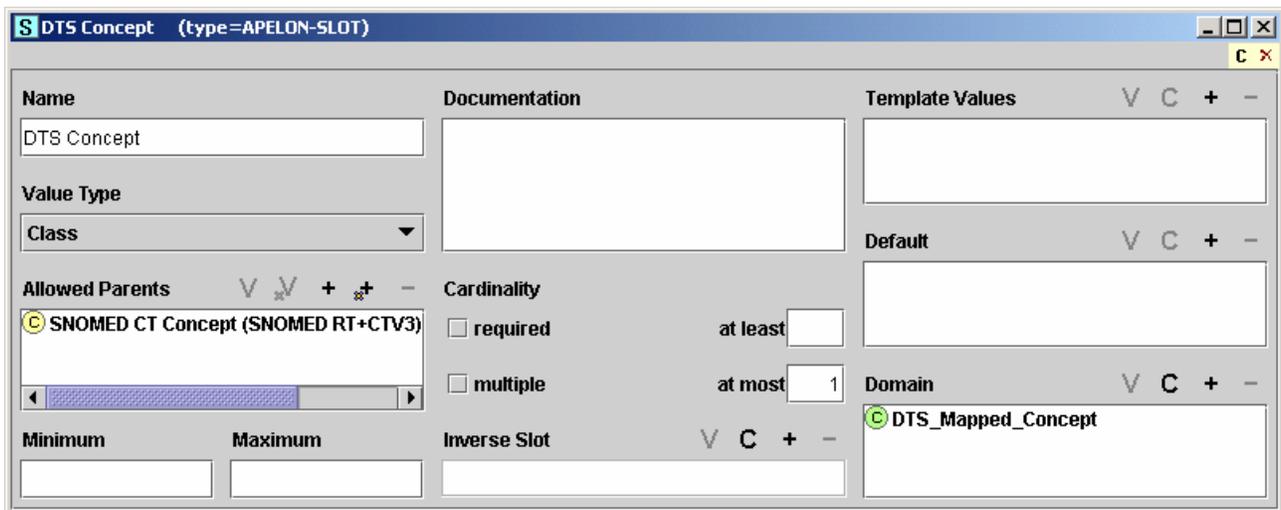
DtsCodedValueSlotWidget

This widget is suitable for CodedValue instances, on the “code” slot. Instead of creating a new CodedValue class as in the previous 2 widgets described, this widget allows lookup of a DTS Concept, and then sets the slot values of that class as described in the section above, in addition to mapping the ancestry of the select concept. Its main purpose is for fixing manually created CVs.

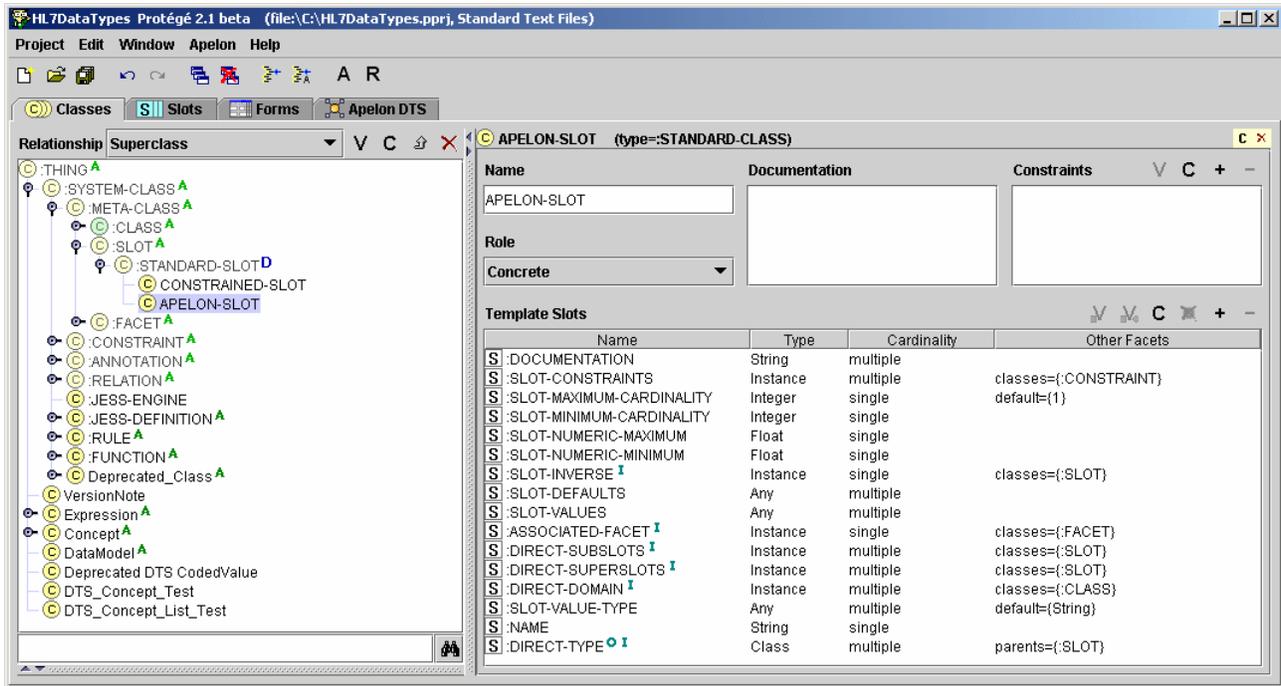


DtsValueTypeWidget

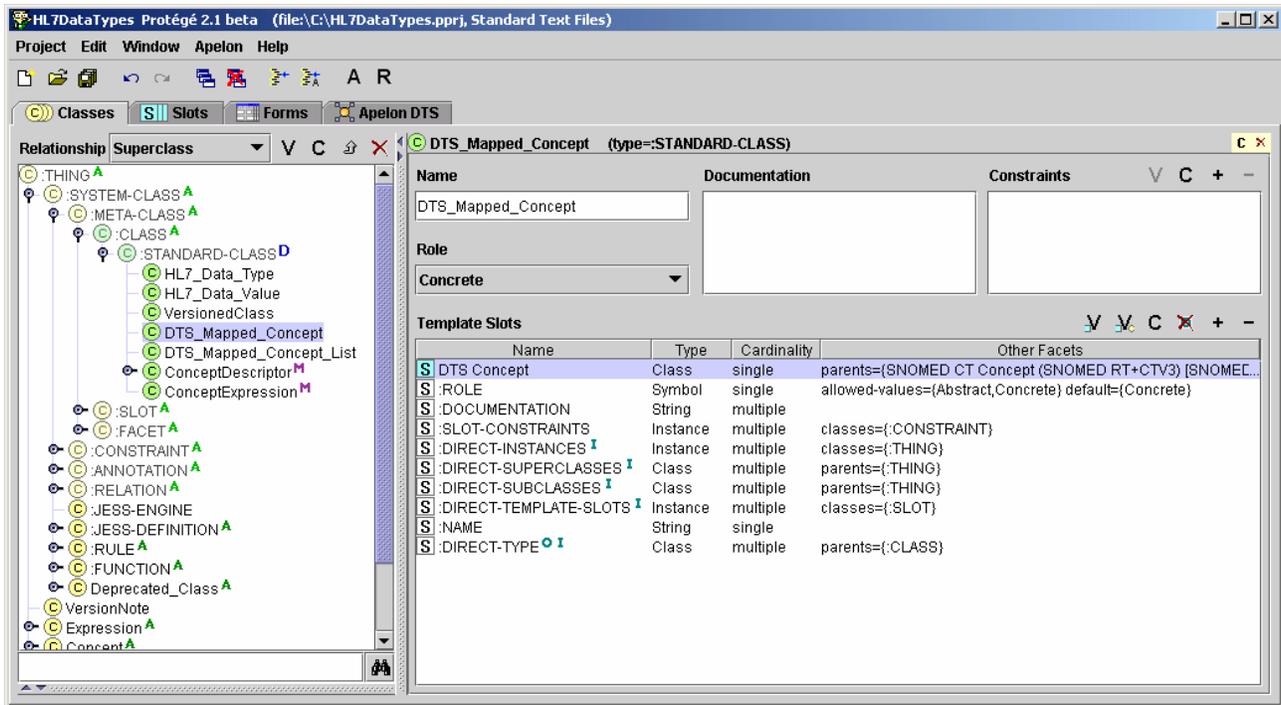
This widget is only suitable for the Value Type facet, and behaves almost exactly like the core Protégé Value Type widget, except, when the value type is set to Class. When the Value Type is changed to “Class”, the “Allowed Parents” widget uses the DtsConceptListSlotWidget, following all the logic of finding, selecting, and mirroring the DTS ancestry of selected concepts.



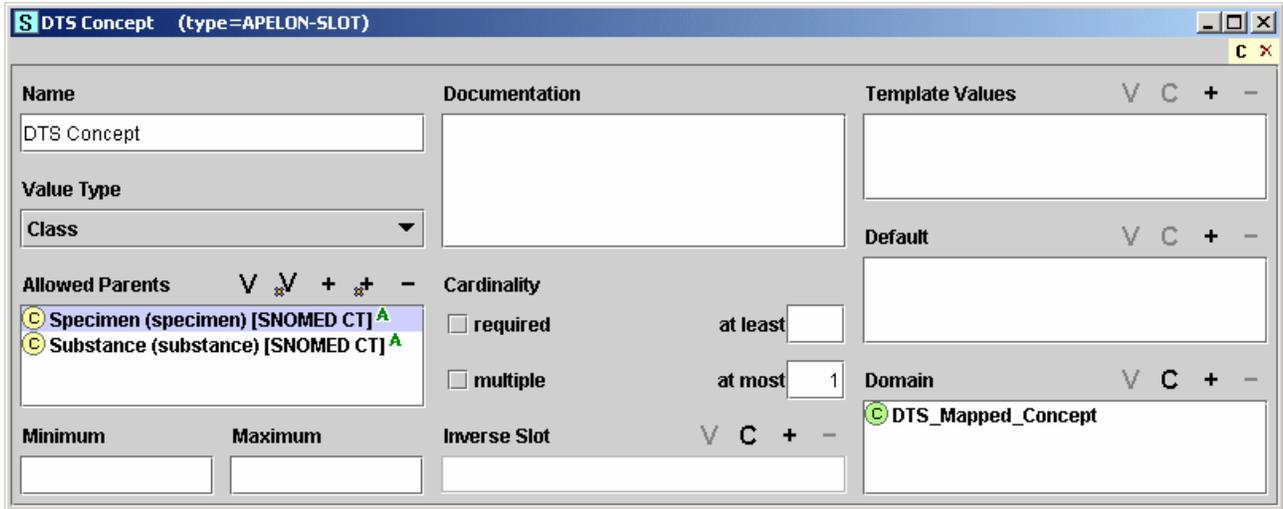
You must change the Form of the Slot itself to use the widget, and since we are not likely to want to change every slot to use the DtsValueTypeWidget, you can create a subclass as illustrated below, mirroring STANDARD-SLOT, no other changes besides the name.



After we have our subplot set up, and changed the form of the Apelon-Slot meta-class so that the ValueType facet uses the DtsValueTypeWidget, we could change the “DTS Concept” slot meta-class to “Apelon-Slot” on our sample “DTS_Mapped_Concept” meta-class. After that is complete, if we go to our DTS_Mapped_Concept meta-class in the classes tab, we will see:



If we view the top level DTS_Concept slot, we can add, view, and remove CodedValues mapped to Apelon DTS Concepts as allowed parents via the usual DTS Concept lookup dialog, in addition to adding regular Protégé classes via the ClsListWidget behavior. As an example, we'll add the SNOMED concepts “Specimen (specimen)” and “Substance (substance)” as allowed parents.



Now, if we go to our sample DTS_Concept_Test concrete class, we can select a value using the DTS Add button. When a selection is made, the widget will perform a check to make sure the DTS Concept is a child of at least one of the “Allowed Parents” we selected earlier (For multiple allowed parents, the result of the check is “ORed” together). If the selected DTS Concept is subsumed by at least one of the Allowed Parents defined in the top level slot, the DTS Mapped CV is created, and the slot value is set. If the selected DTS mapped CV is *not* subsumed by at least one of the Allowed Parents defined in the top level slot, the widget will inform the user that this scenario has been encountered, and give the user the opportunity to override the allowed parent rule.

10. Mapped CodedValue Slot Data

The CodedValue direct instances created by the DTS slot widgets take the following information, the **bold** slot values being the necessary information required to find and retrieve a DTS Concept at a later point in time:

CodedValue

Name	The name of the DTS Concept, and the concept's Namespace name when creating new CodedValues, nothing when changing existing CodedValues via the map utility, or DtsCodedValueSlotWidget
DisplayName	The name of the DTS Concept
Label	The DTS Concept id
Code	The DTS Concept id if SNOMED, otherwise the DTS Concept code
CodeSystemName	The Namespace name where the DTS Concept lives
CodeSystem	An ObjectIdentifier instance created with the below information. Searches to see if one exists before creating

ObjectIdentifier

Name	The Namespace id (We need to keep this for "view" purposes)
Label	The Namespace display name

Note: When running the CodedValue mapping utility on existing CodedValues, the Name field is not changed, since changing the name could invalidate references to the CV from other Frames in the working project, as well as other down stream projects including the working project.

11. CodedValue “Garbage Collection”

Incremental garbage collection

As DTS mapped CodedValue classes are removed, either by explicitly removing them with the “-“ buttons on the widgets or by viewing the selected concept and changing the selection to another concept, the plug-in checks if the removed CodedValue is no longer referenced in the project, and can be “deprecated”. Garbage Collection can be turned on/off by selecting/deselecting the “Apelon>>CodedValue Garbage Collection” check box menu item in the main Protégé menu, which becomes visible once the plug-in has been loaded.

The following rules are used by the plug-in to determine if a CV can be deprecated:

- The CV is not from an included project
- The CV is a leaf in the kb CIs hierarchy (no children)
- The CV has no references to it other than direct subclass reference(s) from its parent(s), and a direct instance reference from its metaclass (CodedValue itself)
- The CV is properly mapped to a DTS Concept

If all of the above rules are met, the leaf CV is removed from the mapped branch of the project, and added to the deprecated branch. After it is moved, the plug-in recursively walks the deprecated

CV's (previous) parents and checks the same rules on each parent, seeing if each parent, that may now be a leaf, can be deprecated.

It is up to the user to actually delete the CodedValues from the deprecated branch of the project.

Project wide CodedValue garbage collection

Users can optionally run CodedValue garbage collection on the entire project by selecting the “Apelon>>Run CodedValue Garbage Collection on entire project” menu item in the main Protégé menu, which again, becomes visible once the plug-in has been loaded. This action follows the same rules as above, first identifying eligible mapped CodedValue leaves, and then working up the kb recursively.

12. CodedValue Mapping Utility

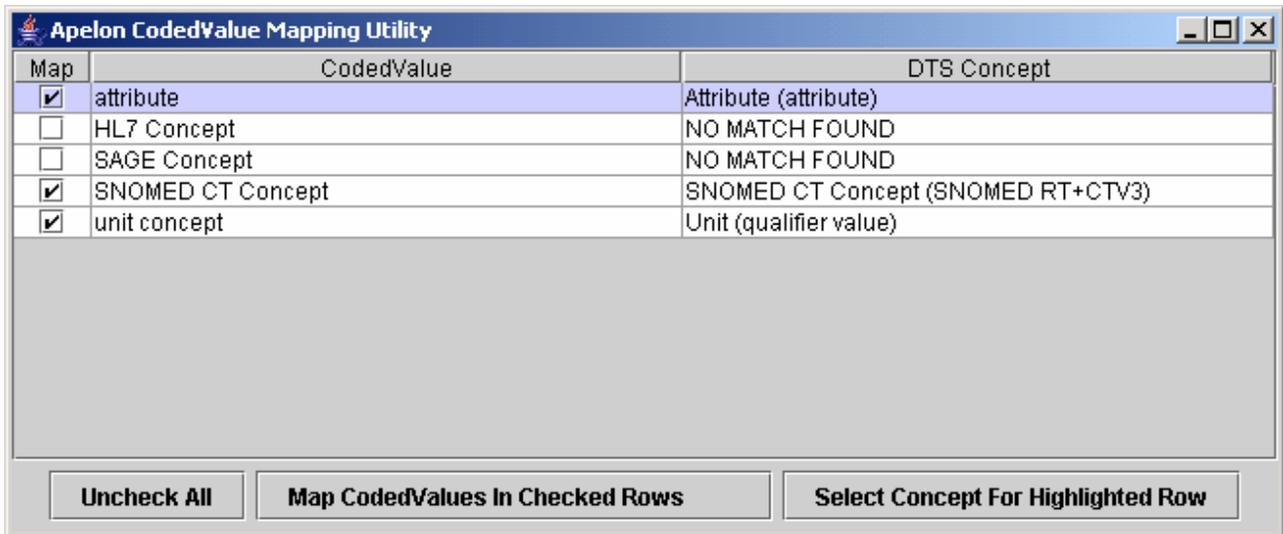
This utility will find each existing CodedValue that does not have the necessary information (refer to the [Mapped CodedValue Slot Data](#) section for details) to find and retrieve a corresponding Apelon DTS Concept.

Running the utility

Once you have opened a project in Protégé, configure the project to use the DtsTabWidget. After you have logged into the DTS Server, select “Apelon>>Map CodedValues...” from the main Protégé menu to work on the CodedValues in the currently open project.

The mapping utility dialog will not come up if the project does not contain the CodedValue metaclass, or, if the project does not contain any unmapped CodedValues. An informational message will be presented so the user knows why mapping cannot be run.

Here is what the mapping dialog looks like when opening the July, 2003 version of HL7DataTypes.pprj from the sageimmunization project:



Upon initial display, the utility:

- Looks for all CodedValues in the selected project and identifies those that do not contain the [necessary info](#) to directly map them to a DTS Concept. Those concepts are placed in the above table.
- Attempts to find matching DTS concepts for the CodedValues not generated by the Apelon Plug-in if the CodedValue contains a numeric value for the "code" slot, and an identifiable DTS namespace name in the "CodeSystemName" slot.
- If the utility finds an exact match based on concept id and namespace name, it checks the "map me" check box next to the CodedValue, and places the name in the "DTS Concept" column, else, if no match is found, "NO MATCH FOUND" is displayed and the check box is left unchecked.

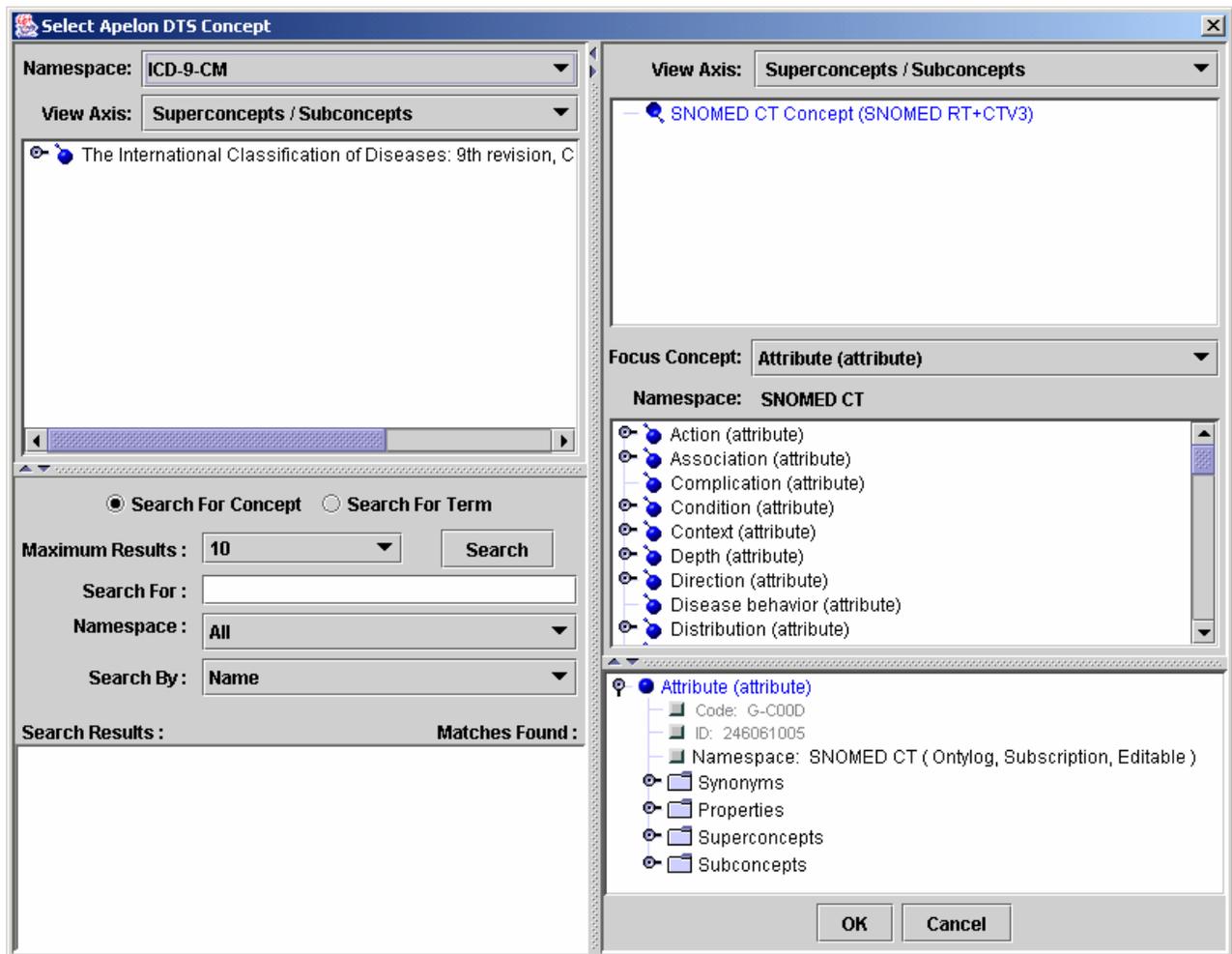
Clicking the "Map CodedValues in Checked Rows" button at this point would attempt to map the checked CodedValues to the found DTS Concept matches, **updating the Protégé .pont file (!)** and outputting the results of the map in the file "Protege_x.x\plugins\CVMapUtil.log". (You may wish to save a backup copy of the project files prior to mapping,)

The mapping operation changes the CodedValue slot values to the appropriate information, as described in the overview. In addition to the CV slot value changes, the utility mirrors the parentage of the DTS Concept into the "special" "Apelon DTS CodedValue" branch of the kb, leaving the "mapped" CodedValue where it is, but also adding it to the mapped branch. (The mapped CodedValue may end up having multiple parents).

Note: For projects containing unmapped CodedValues that reside within included projects, they will be displayed, but the CodedValue column will have “(INCLUDED – NOT MAPPABLE)” appended to the CV name. The check box will not be checkable for these rows, and the “Select Concept for Highlighted Row” button will become disabled when an included CV has been selected. The utility is only able to map CodedValue direct instances that live directly in the opened Protégé project file.

Optional verification and selection of found/unfound matches

The “Select Concept for Highlighted Row” button can be clicked to either inspect the matched DTS Concept, or select a new DTS Concept, for the row selected in the table. This selection dialog also includes the concept walker so you can view the context of concepts that were matched by the utility. For example, if select the "Attribute (attribute)" row, and then click the “Select Concept for Highlighted Row” button, you would see the following dialog; note that the concept walker and view panes have the concept in context, but the namespace navigation and search panels come up in their default state. If you click a button next to a “NO MATCH FOUND”, all panels will come up in their default state.



You could either hit cancel here, and the mapping utility would retain the automatically selected concept, or you could select another DTS Concept, overriding the concept the utility found.

After confirming/changing found matches, and selecting concepts for CodedValues where a match was not found, click the "Map CodedValues" and the utility will go thru the checked CodedValues and add the appropriate information, as well the appropriate hierarchical ancestry underneath the "Apelon DTS CodedValue".

After mapping the selected CodedValues, they are removed from the table, leaving only CodedValues that have yet to be mapped. You can go thru and map selected CodedValues as many times in one session as you would like, the utility will not attempt to remap a CodedValue that has already been successfully mapped.

After you are finished mapping, you can close the dialog, and view changes made to the CodedValues via the regular Protégé UI. The map utility does not save any changes explicitly, so it is up to the user to save the project before closing the project. This provides the ability to simply close a project without saving mapping changes if the mapping results were not what the user expected or desired.

13. Configuration options

All configuration options are stored in the file `/Protege_x.x/plugins/com.apelon.dts_plugin/DTSPugin.xml`. Most are related to connection, and not meant to be edited. The following four properties can be changed:

```
<property name="rootImportCls" value="Concept" />
```

The `rootImportCls` property determines the root Class under which to place all the DTS mapped CodedValue direct instances. While this is configurable, we recommend agreeing on a root ASAP and sticking with that root, otherwise we are going to end up with a maintenance mess.

```
<property name="rootDeprecateCls" value="Deprecated DTS CodedValue" />
```

The `rootDeprecateCls` property determines which Class all deprecated DTS mapped CodedValues will be placed under.

```
<property name="garbageCollect" value="true" />
```

Determines whether or not to perform DTS mapped CodedValue garbage collection on individual DTS concept slot widget remove actions. Users can configure this by way of the "Apelon" menu in the main Protégé menu.

14. Troubleshooting / FAQ

I can't connect to the SAGE DTS Server

If you cannot connect to the DTS server on port 2323, try the following DOS commands to troubleshoot:

If you can't ping it:

```
ping 63.126.41.231 <enter>
```

you most likely can't get out to the Internet.

If you can ping, but still not connect, try the following:

```
telnet 63.126.41.231 2323 <enter>
```

A "successful" connection will actually result in a blank black screen where you can not see the chars you type, just close the window by hitting the "x" at top right. An error message will be displayed if telnet could not connect. An error message most likely points to firewall issues, check with your network administrator to ensure you can get out on port 2323, over TCP.