**Lego Editor User Guide**

**U.S. Department of Veterans Affairs**

**March 26, 2013**

**Blanket Purchase Agreement 54 (VA776-11-BP-0054)**

**Task Order 4 Lego Software Tools**



Table of Contents

[1. Introduction 4](#_Toc352076118)

[1.1. LEGO 4](#_Toc352076119)

[1.2. Snomed Observables Model 5](#_Toc352076120)

[2. Installation 6](#_Toc352076121)

[2.1. Supported Platforms and Requirements 6](#_Toc352076122)

[2.2. Installing 6](#_Toc352076123)

[3. Function Overview 8](#_Toc352076124)

[3.1. Menus 9](#_Toc352076125)

[3.1.1. File Menu 9](#_Toc352076126)

[3.1.2. Edit Menu 10](#_Toc352076127)

[3.1.3. Help Menu 10](#_Toc352076128)

[3.2. Left Panels 11](#_Toc352076129)

[3.2.1. Lego Lists 11](#_Toc352076130)

[3.2.2. Snomed Browser 12](#_Toc352076131)

[3.2.3. Snomed Search 13](#_Toc352076132)

[3.2.4. Templates 15](#_Toc352076133)

[3.2.5. Pending Concepts 16](#_Toc352076134)

[3.3. Editor Area 17](#_Toc352076135)

[4. Lego Lists Panel 19](#_Toc352076136)

[4.1. Context Menus 19](#_Toc352076137)

[4.2. Filtering 22](#_Toc352076138)

[4.2.1. Advanced Filtering 23](#_Toc352076139)

[4.3. Entering Concepts in the Concept Filtering Fields 24](#_Toc352076140)

[5. Editor 25](#_Toc352076141)

[5.1. Context Menus 25](#_Toc352076142)

[5.2. Concept Editing 27](#_Toc352076143)

[5.3. Save and Versioning 29](#_Toc352076144)

[5.4. Supported Drag and Drop 30](#_Toc352076145)

[5.5. Error Display 31](#_Toc352076146)

[5.6. Pending Concepts 31](#_Toc352076147)

[5.7. Advanced Editing 32](#_Toc352076148)

[5.7.1. Infinite Nesting 32](#_Toc352076149)

[5.7.2. Conjunctions 32](#_Toc352076150)

[5.7.3. Relationship Groups 33](#_Toc352076151)

[5.7.4. Measurements 34](#_Toc352076152)

[6. Lego Exporting Options 35](#_Toc352076153)

[6.1. Exporting 35](#_Toc352076154)

[Appendix A: Installed Files Overview 36](#_Toc352076155)

[Appendix B: Lego XML Schema 37](#_Toc352076156)

# Introduction

The Lego Editor was developed as a toolset which will be an extension of the IHTSDO Workbench, for use in building reusable clinical information models based on standard terminology. Conformant with the modeling framework known as Lightweight Expression of Granular Objects (LEGO) and the principles of the Simple Integrated Model (SIM), these models can help create a robust, semantically interoperable mechanism for creating and sharing electronic health data

## LEGO

Lightweight Expression of Granular Objects (LEGO) is a strategy to capture complex clinical statements by combining small information units based on standard terminologies. LEGOs are thus conceptually similar to Lego® building blocks, in that small standardized pieces can be combined to create large, complex sculptures. LEGOs are themselves based on the Simple Integrated Model (SIM). Although many standard terminologies exist for different domains, LEGOs rely heavily on SNOMED CT, which is emerging as the de facto standard for structured clinical documentation because of its wide domain coverage and formal semantic structure. The LEGO framework describes a set of rules to represent simple and complex clinical statements. Relying on the SNOMED CT logical structure makes it possible for a computer system to compare different LEGOs and determine how they are related. Key components of the LEGO are listed below.

* Discernible: A discernible is something that is perceptible by the senses or intellect. Negation within a discernible is not allowed and conditions support the negation of an entire discernible. A discernible is represented as an encoded SNOMED expression, represented by a single code after classification.
* Qualifier: A qualifier serves to qualify a value. For example, a qualifier of "unknown, not asked" applied to a value of "null" makes the meaning of "null" more specific. This qualification allows representation of the HL7 null flavors, in addition to allowing a means to qualify substantive values with qualifiers such as "default," "patient entered" and "confirmed default."
* Timing: Timing is a representation of the asserter's belief in the timing during which this assertion is valid. Timing can be represented as a point, a bound or an interval.
* Value: The value of a condition can be of several types. The first is numeric, which represents a point, bound or interval. An example of a numeric value is a patient’s pulse. Values may also be represented as an encoded expression or a Boolean, such as true or false. However, values must not be used to represent modifiers such as “severe” or “green,” which would instead be a part of the Discernible expression.

## Snomed Observables Model

The IHTSDO is currently developing a new model for representing observable entities. The primary documentation for this model can be found in the IHTSDO document entitled SNOMED CT Style Guide: Observable Entities and Evaluation Procedures (Laboratory). Many concepts that currently exist in the SNOMED CT "Clinical findings" hierarchy will eventually be moved to an expanded observables hierarchy. In an effort to avoid obsolescence as the new observable model is implemented in future releases of SNOMED CT, it was decided that the LEGO project should use this new model to represent PNCS content. This decision can be understood from the perspective that PNCS items are can be viewed as observations recorded on data collection forms.

The observables model is intended to represent what is observed and how the observation is made. The design of the model has been influenced by work on the Basic Formal Ontology (BFO), which is described at <http://www.ifomis.org/bfo/>. Of particular note is the idea that the target of observation is often a quality or PROPERTY TYPE (such as color or length), which INHERES IN an “independent continuant” (such as a substance, body structure, or physical object). An "independent continuant” is an entity that persists through time and does not depend on any other entity for its existence. Thus, "independent continuants” are the typical objects that we see around us every day.

An observable entity concept is modeled by its specifying attributes, each of which has a set of SNOMED CT concepts as its allowable values. The attributes used in the observables model and their corresponding allowable values are defined in the SNOMED CT Style Guide: Observable Entities and Evaluation Procedures (Laboratory) document, and a user of the LEGO Editor should become familiar with this document before attempting to model PNCS content. Some of the attributes in the observables model (such as PROPERTY TYPE, INHERES IN, and CHARACTERIZES) describe the intended target of observation, while others (such as SCALE, UNITS, and DIRECT SITE) describe the act of observation (i.e., how the observation was made). For example, the observation of core body temperature by means of an ear thermometer can be represented as shown below. The DIRECT SITE attribute specifies the "proxy" location of the observation. (In other words, the intent is to observe the core body temperature, but the observation is actually made by measuring the temperature in the ear.)

|Observable entity|:

* 1. |PROPERTY TYPE| = |Temperature|,
  2. |INHERES IN| = |Body internal region|,
  3. |UNITS| = |degrees C|,
  4. |DIRECT SITE| = |Tympanic membrane structure|

The discernible is the part of a LEGO assertion in which PNCS content is represented by an expression based on the observables model. In the LEGO Editor, this expression begins with an observable entity concept, which is then specialized by adding relationships. Each relationship consists of a type (which is an attribute defined in the observables model) and a value (which is one of the allowable values for that particular attribute). Because the discernible will be processed by an enhanced Description Logic classifier that can handle numeric values, it is often desirable to represent the PROPERTY TYPE implicitly. In the example shown above, this would mean using a relationship where the type is "Temperature" and the value is a measurement, such as 37.5 degrees Celsius. The other parts of a LEGO assertion (value, qualifier, and timing) can be regarded as being outside the scope of the observables model documented in the Style Guide.

# Installation

The Lego Editor installation files are hosted on the Veterans Administration Project Collabnet website, which is hosted by the IHTSDO.

<https://csfe.aceworkspace.net/sf/projects/veterans_administration_project/>

You will need a Collabnet account and the appropriate permissions to access the installers.

The Lego Editor is published in the following release folder:

<https://csfe.aceworkspace.net/sf/frs/do/listReleases/projects.veterans_administration_project/frs.lego_editor>

## Supported Platforms and Requirements

As a Java program, the Lego Editor supports most common operating systems. Installers are provided for Windows, OSX and Linux. All dependencies of the editor are included with the installer distributions – including the Java Runtime and the Snomed Database.

It is highly recommended that the Lego Editor be run on a 64bit operating system, with at least 4 GB of RAM. 8 GB or more of RAM is preferred. While it can be run on a 32 bit operating system, performance will be poor due to memory limits.

## Installing

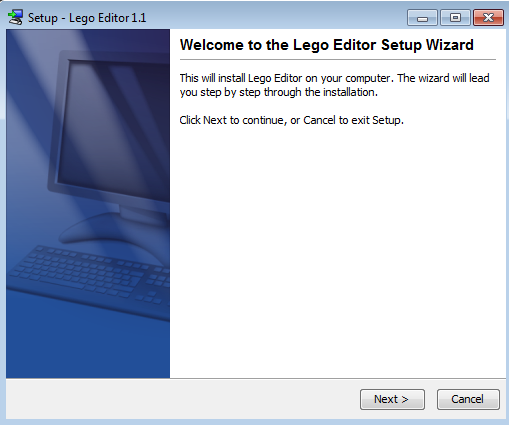
The current release of the Lego Editor provides the following installers:

* Lego-editor-1.1-installer\_x64.dmg – Mac OSX installer
* Lego-editor-1.1-installer\_x64.exe – Windows 64 bit installer
* Lego-editor-1.1-installer\_x64.sh – Linux 64 bit installer
* Lego-editor-1.1-installer\_x32.exe – Windows 32 bit installer (not recommended)

Download the appropriate installation file for your system.

The installer should be launched with administrative level privileges. When the installer starts, a wizard will step you through the installation process. See Figure 1.

Figure



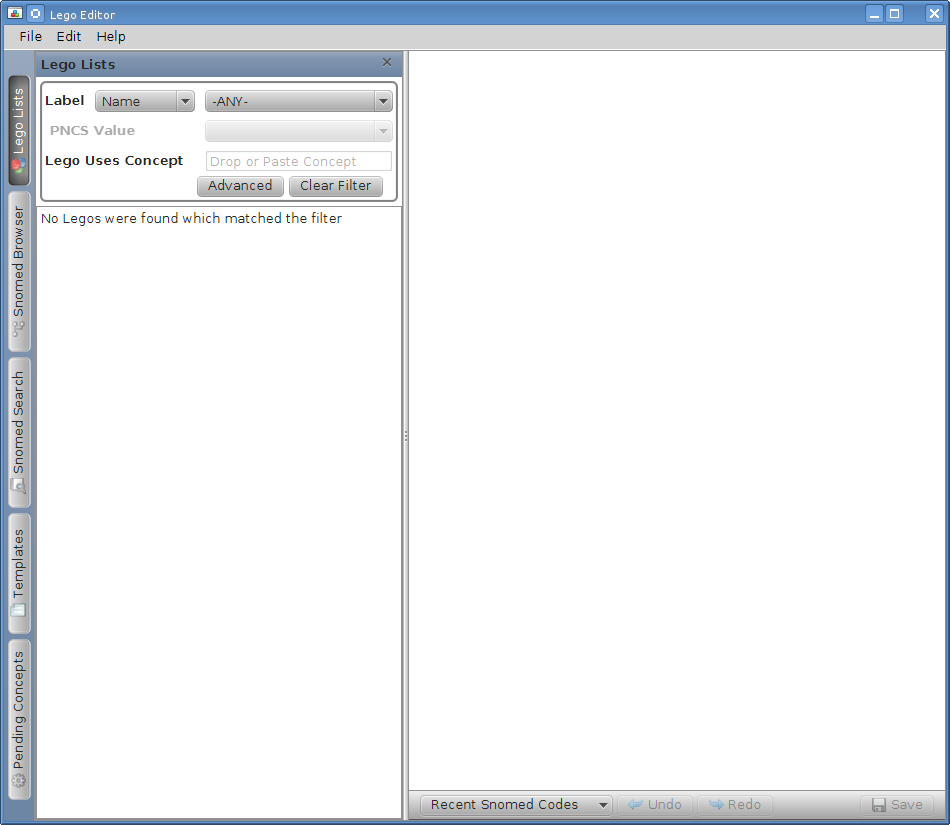
The editor only asks standard questions about creating platform shortcuts during the installation.

Note: On Windows, it is recommended that you use the default installation path, and not a path in the “Program Files” system folder.

# Function Overview

When the Lego Editor is launched, you will be presented with the Lego Editor GUI, as shown in Figure 2. This section of the user guide will give a brief overview of each of the functional areas of the Lego Editor.

Figure

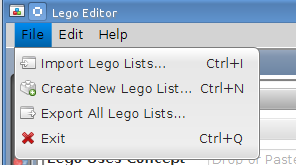


## Menus

There are three primary menus in the Lego Editor. File, Edit, and Help.

### File Menu

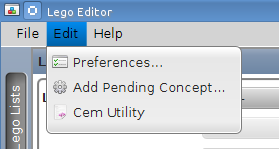
Figure



* Import Lego Lists – This function allows the user to import Lego XML files into the Lego Editor for viewing or further editing. The format of the XML files that may be imported is described in Appendix B: Lego XML Schema. Any number of Lego files can be imported at once. Typically, you would use this to import Lego XML files that were created by another installation of the Lego Editor.
* Create New Lego List – This function displays the dialog for creating a new Lego List within the Editor.
* Export All Lego Lists – This function is used to display the Export Dialog, which is further described in Lego Exporting Options.
* Exit – Closes the Lego Editor.

### Edit Menu

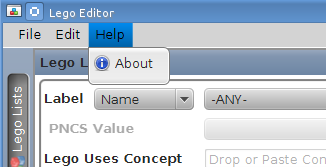
Figure



* Preferences – This menu brings up the Preferences Dialog which allows you to specify the following program options
  + Author Name – The name to assign to newly created LEGOs.
  + Module – The Module to assign to newly created LEGOs.
  + Path – The Path to assign to newly created LEGOs.
  + Summary View – Enable or disable the Lego Summary View window in the Editor.
  + Description Type – Display “Fully Specified Name” descriptions from Snomed, or “Synonym Preferred”
* Add Pending Concept – Display the dialog that allows the creation of new Pending Concepts, which are discussed in more detail in Pending Concepts.
* Cem Utility – Display the CEM import tool, which can be used to create or find LEGOs that model items from a CEM CDL XML file.

### Help Menu

Figure



* About – Displays the program version.

## Left Panels

The Lego Editor has 5 buttons down the left edge of the editor window. Each of these buttons shows or hides a different feature of the editor. The buttons can be enabled or disabled in any combination.

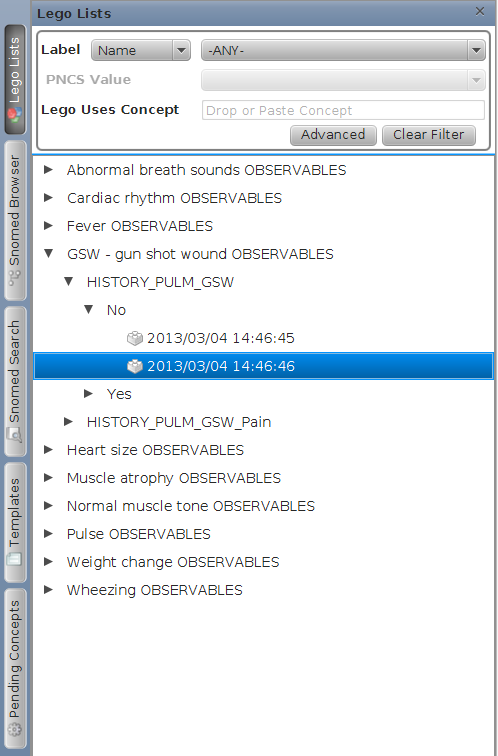
In Figure 2 above, the “Lego Lists” button is enabled, and the Lego Lists window is showing in the left third of the Editor.

When all buttons are disabled, the left split of the Lego Editor disappears entirely. The size of the left panel can also be adjusted by sliding the divider.

### Lego Lists

The Lego Lists panel shows all of the Lego Lists (and contained LEGOs) that are present in the Lego Editor. There are a number of filtering options that can be used to reduce the displayed set of LEGOs – which are discussed in detail in section 4.2 below.

Figure



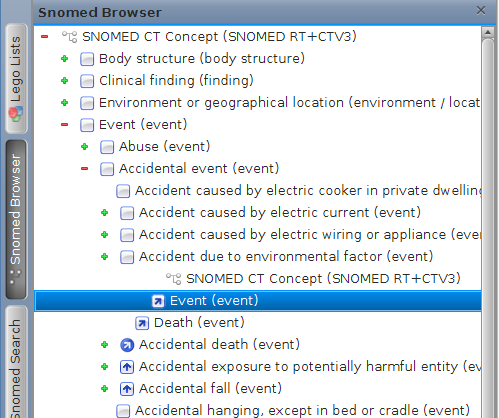
.

LEGOs are organized in the Lego Lists panel with a short hierarchy. At the top of the hierarchy is the Lego List name. The next item in the tree is the PNCS Name which a given Lego models. The next level in the tree is the PNCS Value which a given Lego models. Finally, the individual LEGO(s) for the specified PNCS Name and PNCS Value are listed.

### Snomed Browser

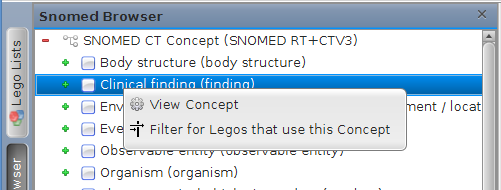
The Snomed Browser panel is used to browse the Snomed Taxonomy.

Figure



The browser supports navigating down the tree, and also showing the alternate path back to root when concepts have multiple locations within the Snomed taxonomy, as shown above. You can also drag and drop concepts from the tree to other fields within the Editor. Each item in the tree has a context menu that appears if you right or alt click on the item.

Figure

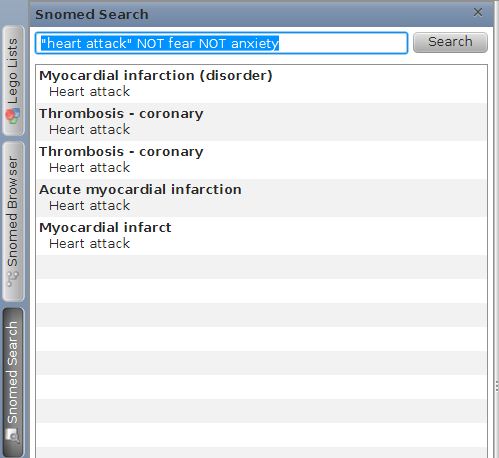


* View Concept – Displays a detailed concept viewer window for the specified concept
* Filter for LEGOs that use this Concept – Constructs a concept filter in the Lego Lists panel.

### Snomed Search

The Snomed Search panel is used to search the descriptions of Snomed for concepts that match the query criteria.

Figure

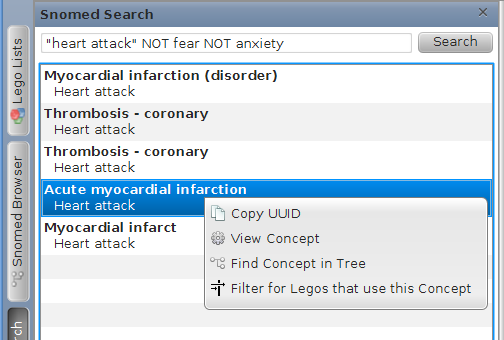


The Search engine supports double quotes for phrases, operators such as ‘AND’ or ‘NOT’. For full documentation on the types of advanced queries that can be written, see: <http://lucene.apache.org/core/4_2_0/queryparser/org/apache/lucene/queryparser/classic/package-summary.html#Overview>

Each of the displayed results shows the preferred description of the concept in bold, followed by the description(s) that matched the query. Each result supports drag and drop to other fields within the editor.

The results also have context menus:

Figure



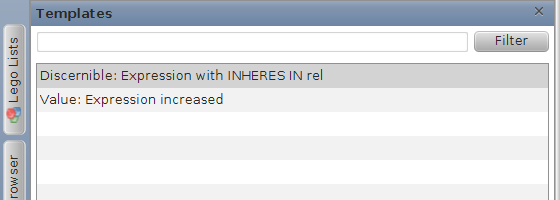
* Copy UUID – Places the UUID for the selected concept on the system clipboard
* View Concept – Displays a detailed concept viewer window for the specified concept
* Find Concept in Tree – Locates the selected concept in the Snomed Browser.
* Filter for LEGOs that use this Concept – Constructs a concept filter in the Lego Lists panel.

### Templates

The Templates panel shows all user created templates that are currently loaded in the Lego Editor. A template can be created by a user from any existing LEGO, or from a component of an existing LEGO via context menus.

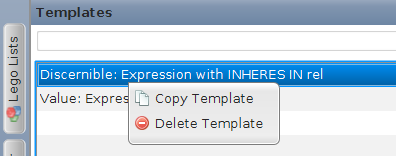
For example, if you need to model 10 different LEGOs that all have the same discernible, you can create the first LEGO, then use that LEGO to create a template. Future LEGOs can then be created from the Template.

Figure



By default, all of the templates that are currently in the Lego Editor are displayed. This list can also be filtered with a simple substring search.

Figure



The context menus on templates are

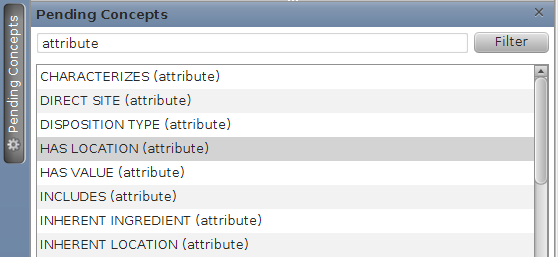
* Copy Template – Put the template on the system clipboard for use in the editor.
* Delete Template – Remove the selected template from the system. Note, this has no impact on any LEGOs that may have been created with the template.

The template panel will also display a tools tip which shows a summary view of the template when the mouse is hovered over the template.

### Pending Concepts

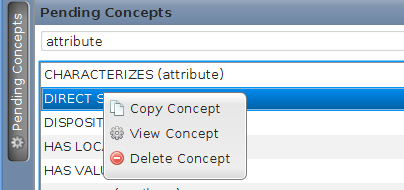
The Pending Concepts panel is used to display concepts that have been manually created for the purpose of authoring LEGOs – discussed in more detail in section 5.6.

Figure



The Pending Concepts panel supports sub-string filtering, as shown above. Each item in the panel supports drag and drop to other fields within the editor, and also has the following context menus:

Figure



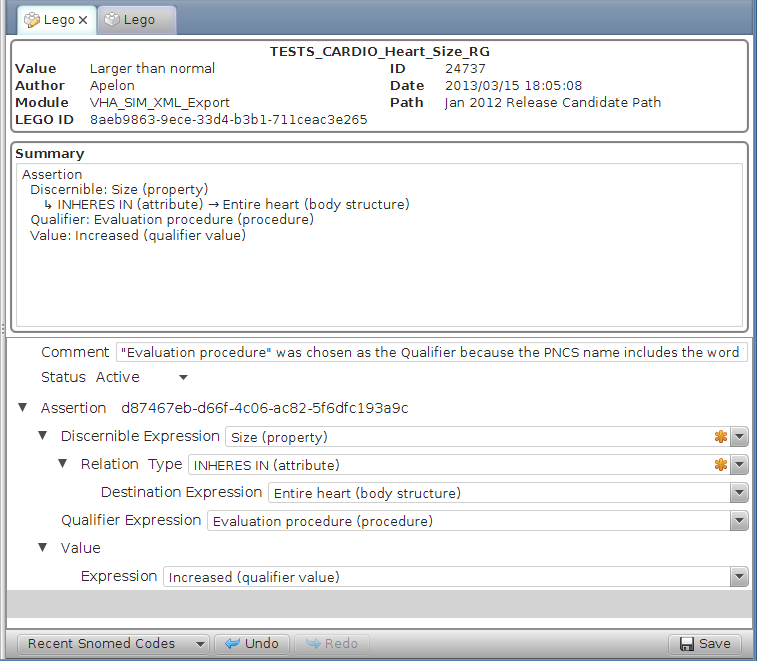
* Copy Concept – Copies the concept on to the system clipboard.
* View Concept – Displays the concept in the Pending Concept dialog. Note – Pending Concepts cannot be edited after they have been created.
* Delete Concept – Deletes the Pending Concept from the Lego Editor. Note – this does not modify any LEGOs that have references to the pending concept. In the future, those LEGOs will simply show that they contain an unknown concept.

## Editor Area

The Editor Area is the right 2/3s of the screen as shown in Figure 2. Note that in Figure 2, the right 2/3s of the screen is empty, because no LEGOs are open for editing.

When LEGOs are open, the view will show a tabbed viewer, with one Lego per tab.

Figure



Tabs will contain a “pencil” icon when the LEGO within has unsaved changes.

Working from top to bottom – the first section of the Editor window shows the PNCS and Lego ID information.

* TESTS\_CARDIO\_Heart\_Size\_RG – This is the PNCS Name
* Value – The PNCS value that is being modeled.
* ID – The PNCS ID tied to the PNCS Name.
* Author – The author that created this LEGO
* Date – The date and time when this LEGO was stored.
* Module – The module assigned to this LEGO
* Path – The Path assigned to this LEGO.
* Lego ID – The generated UUID to identify this LEGO.

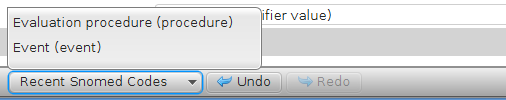
Some of these fields can be specified in the Edit Menu “User Preferences’. The PNCS values are specified in the Lego creation dialog. The rest of the values are automatically generated by the Lego Editor.

The next section contains a textual summary view of all of the information in the Lego.

The third section contains the actual Lego Editor. This will be discussed in section 0.

Finally, the bottom row of the Editor area contains number of buttons:

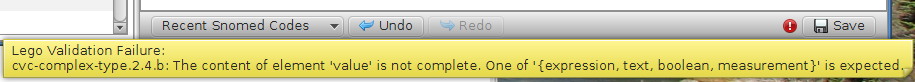
* Recent Snomed Codes – Shows the most recently used Snomed Concepts. Clicking on the button shows a menu that allows you to drag and drop a concept into the editor.   
  **Figure 16**



* Undo – Undo the most recent change to the currently displayed LEGO
* Redo – Redo the most recent undone change to the currently displayed LEGO
* Save – Store the LEGO that is being edited – creating a new LEGO. The previous state of the LEGO is not changed. The Save button is only enabled when the Lego has been changed from the Lego that is currently stored. See Save and Versioning, below.

The bottom row of the Editor area also may contain an error marker next to the Save button.

Figure



This marker appears when the LEGO that is currently being displayed contains an error that would cause the Lego XML file to fail validation against the Appendix B: Lego XML Schema. Hovering the mouse over the red error indicator will display the error message.

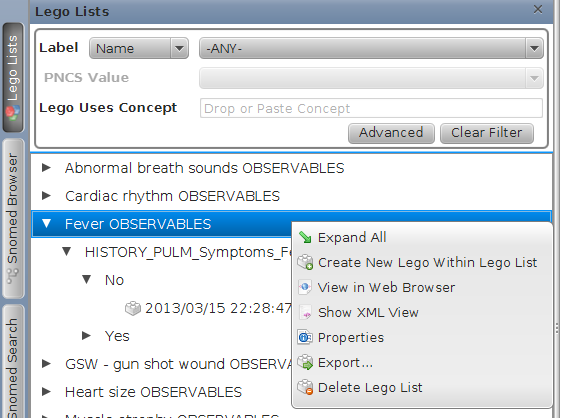
# Lego Lists Panel

The Lego Lists panel was briefly described in section 3.2.1, where the general purpose of the panel and the hierarchy of the tree is described.

## Context Menus

Each level of the Lego List hierarchy has a different context menu that will appear upon right or alt click. Starting with the top, or Lego List node:

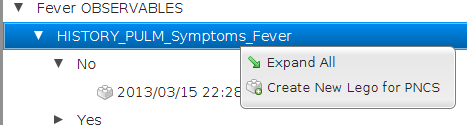
Figure



* Expand All – Opens all tree nodes from the currently selected node down.
* Create New Lego With Lego List – Opens the Create New Lego dialog box.
* View in Web Browser – Displays a web-transformed view of the selected Lego List in the default web browser installed on the system.
* Show XML View – Displays the XML formatted rendering of the selected Lego List.
* Properties – Displays the Lego List dialog, which can be used to edit the Name, Description, and Comment fields for the specified Lego List.
* Export – Launches the Lego List export dialog so you can specify how to export the selected Lego List. Also see Exporting.
* Delete Lego List – Delete the currently selected Lego List, and all LEGOs contained within it.

Next, we have the PNCS Name node.

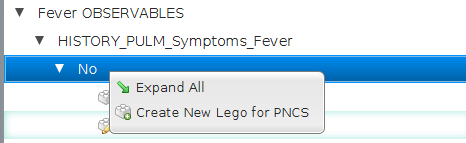
Figure



* Expand All – Opens the tree nodes from the selected node all the way to the bottom of the tree.
* Create New Lego for PNCS – Opens the Create Lego dialog with the PNCS Name and ID pre-populated.

On the PNCS Value node, we have the same menu options as on the PNCS Name node.

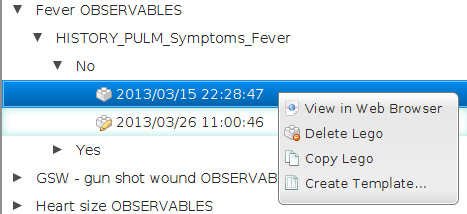
Figure



However, in this case, if you select the “Create New Lego for PNCS” option, no dialog will be displayed – a LEGO will simply be created, as all of the PNCS information is known based on the location of the click.

Finally, on the individual LEGOs, we have the following options:

Figure

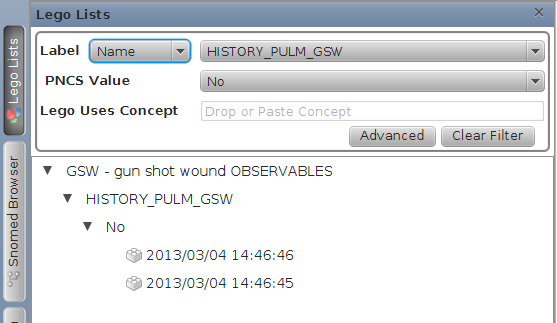


* View in Web Browser – Launch the system Web Browser, and display a web formatted version of just the selected Lego
* Delete Lego – Removes the selected Lego from the Lego List – deleting it from the system.
* Copy Lego – Copies the selected Lego onto the system clipboard. When a Lego is present on the clipboard, it can be pasted onto another Lego List, or onto another PNCS Value – which will cause the Lego to be copied with the appropriate changes to the PNCS entries.

## Filtering

The Lego Lists panel shows all Lego Lists in the system by default. In order to restrict the view, the panel supports filtering by a number of different criteria.

Figure



Shown above is a filter which only displays LEGOs which have a PNCS Name of “History\_PULM\_GSW” and a PNCS Value of “No”.

Filter Options

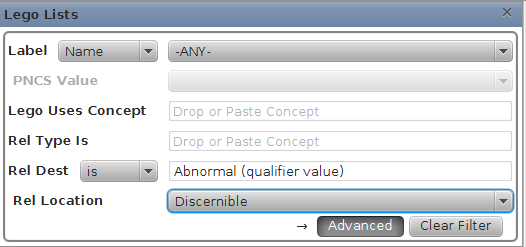
* Label – This line allows you to specify which PNCS name or ID that you wish to restrict to. By changing the dropdown that currently says “Name” you can choose to specify the PNCS by Name or ID. The content of the right dropdown will change to match the selection of the left – displaying either Names or IDs.
* PNCS Value – This line allows you to further restrict the LEGOs that match the filter by specifying the PNCS Value that the LEGOs must contain. This filtering option is only available after PNCS Name or ID filter is applied.
* Lego Uses Concept – This field allows you to specify a Snomed Concept that must be present within the Lego in order for the Lego to match the filter. If a concept is placed here, it may occur at any point within the Lego – but it must appear at least once.
* Advanced – The Advanced Button displays the advanced filter fields, described below.
* Clear Filter – Removes all filter options – causes the panel to display all LEGOs.

Note: Filters only apply to LEGOs that have been saved at least once. If a new Lego has been created that has not yet been saved, it will always appear in the Lego List.

### Advanced Filtering

When the Advanced button is selected, the following additional filter fields are displayed.

Figure



* Rel Type Is – This concept field allows you to specify that a LEGO must contain a particular Snomed Concept as the “Type” of a Relationship.
* Rel Dest – This concept field allows you to specify that a LEGO must contain a particular Snomed Concept as the “Destination” of a Relationship. The drop down that currently says “is” can be set to “is” or “Child Of”
  + “is” – the LEGO must contain this exact concept as a “Destination”
  + “Child of” – the LEGO must contain this exact concept, or any child of this concept.
* Rel Location – This option allows you to specify that the specified relation concepts defined in the previous two fields must occur within a specific location within the LEGO. This field can only be used when at least one of the previous fields is in use.

The Advanced filter options apply even if they are not currently visible. An arrow appears beside the Advanced button to indicate when there are filter criteria present in the Advanced area.

## Entering Concepts in the Concept Filtering Fields

There are a number of ways that you can enter Snomed Concepts into the three different concept fields in the Lego Lists filter options.

* Drag and Drop – Concepts can be dropped into these fields from any other place in the editor where Snomed concepts are displayed.
* Copy / Paste – Concepts can be copied and pasted in to these fields from any other place in the editor where Snomed concepts are displayed.
* ID Entry – UUIDs or SCTIDs can be placed in this field.
* Typing / Quick Search – Typing in the field will cause a Snomed description search to begin, and allow you to select a concept from a pop-up window, as shown below.

Figure

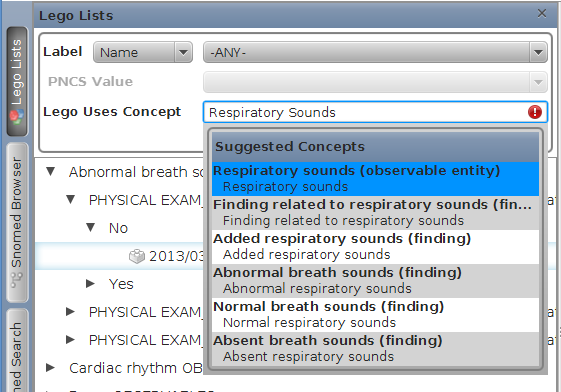


Figure 21 shows the quick search window being displayed. Note that Figure 24 also shows a red error marker that will appear on a concept field any time the current entry is not a valid concept. Entries that are not valid concepts will not be used during the filter execution.

Hovering the mouse over the error icon will show more details about the error.

# Editor

The Editor section of the Lego tool is a tree view of the Lego which enforces the structure required by the Appendix B: Lego XML Schema. All Creation and Removal of Lego components are done via right-click or alt-click context menus. Snomed Concepts are placed within Concept Nodes in the tree via drag and drop, copy paste, or via a quick search.

The context menus that are displayed on a particular element are specific to the element, and the current state of the element. Menus will not appear that allow you to create an invalid LEGO.

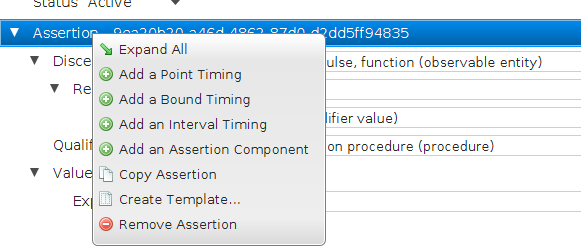
The tree also does content specific validation of the entries – ensuring that all concept fields refer to a valid concept, and that no content specific rules are violated, such as using a concept that is not a child of “Observable” for the Discernible concept.

## Context Menus

Context Menus drive most of the editing process, and dynamically change based on the current state of the LEGO.

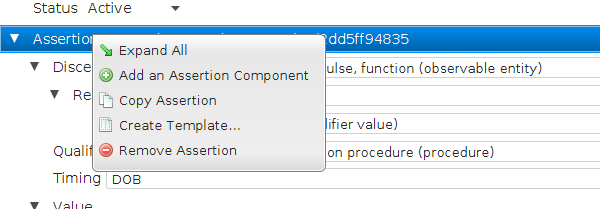
In the following example, the displayed LEGO does not yet have a Timing field. This causes the context menu on Assertion to present a number of options for creating a Timing element, as shown below.

Figure



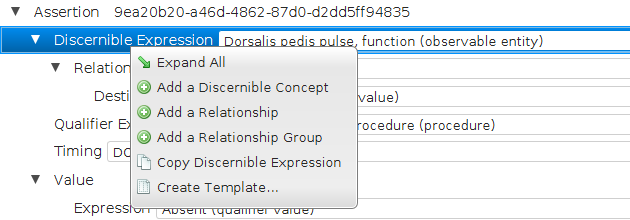
After creating a Timing within this LEGO, the context menu on Assertion will no longer display “Add Timing” options, as shown below.

Figure



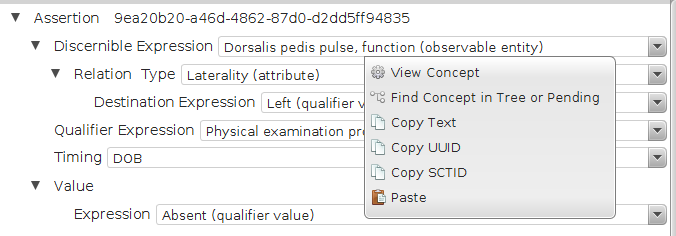
There may also be multiple context menus within a single row of the tree. For example, here is the context menu on “Discernible Expression”.

Figure



And here is the context menu that appears when you right click or alt-click on the concept editor field shown on the same row as Discernible Expression

Figure



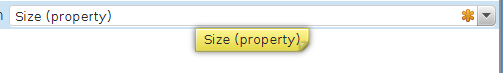
## Concept Editing

Snomed Concept fields in the editor support many features. As shown in Figure 28, each concept field has a context menu which supports the following commands:

* View Concept – Displays all concept details in a concept viewer window
* Find Concept in Tree or Pending – Will locate the current concept in the Snomed Browser or Pending Concepts viewer, depending on the concept type.
* Copy Text – Copy the currently displayed text onto the system clipboard
* Copy UUID – Copy the UUID for the currently selected concept onto the system clipboard
* Copy SCTID – Copy the SCTID for the currently selected concept on to the system clipboard
* Paste – Paste the text from the system clipboard into the concept editor – which will then try to find the appropriate concept

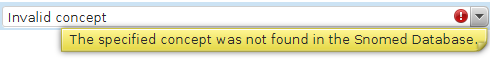
The Concept Editor will display a description from the concept that is selected. The type of Snomed description that is shown is configurable in the User Preferences within the Edit Menu.

Figure



Above, there is an asterisk on the right edge of the concept. This indicates that the concept is a Pending Concept. Also, hovering the mouse over the concept will display a tooltip with the concept text – which is useful when the text string is too long to fit in the editor area.

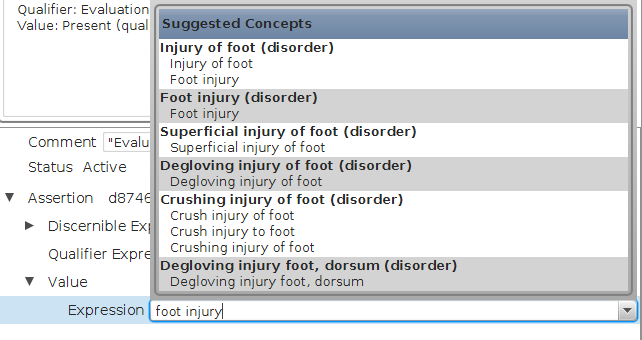
Figure



In Figure 30, the concept editor is displaying a red indicator to show that the currently entered value cannot be resolved to a valid Snomed Concept. Hovering the mouse over the indicator will provide a tooltip with a detailed explanation for the error.

You may also start typing in a Concept field. As soon as the first character is entered, a search will begin for concepts that match the specified words. The more letters and words that you type, the more exact the match will be.

Figure

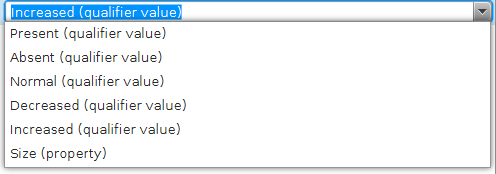


In Figure 31 you can see the search results that are displayed as a result of typing “foot injury”. You may select a Snomed concept from the popup window at any time. While popup window is busy finding results, a progress bar is displayed at the top. When the search is complete, the window appears as it does above. Pressing Esc will close the popup window, as will clicking on another area of the editor.

The search algorithm used for the popup window requires that every entered word be present within a description of the concept – and the last word entered is treated as a prefix match.

Finally, each Concept Editor has a drop down field.

Figure

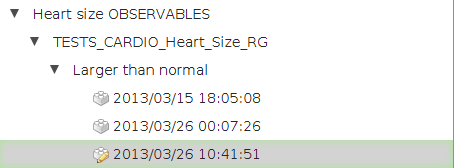


The drop down will display up to 10 concepts. Five of the concepts are the most used concepts for this field within the LEGOs that are loaded in the editor. The other 5 concepts will be the most recent concepts that have been used for this field during this editing session.

## Save and Versioning

Each time a Lego Edit is saved, a new version of the Lego is created.

Figure



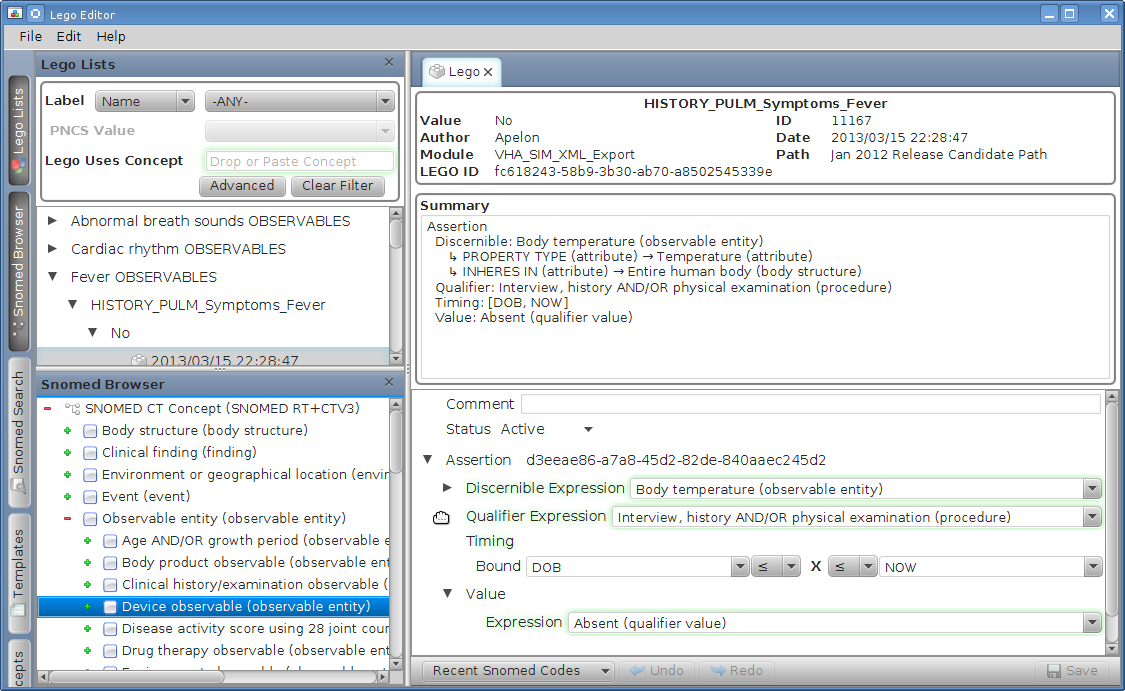
In the image above, you can see 3 different versions of the same Lego that have been saved. Old versions can be deleted by the user. The Pencil icon appears when a Lego has changes in the editor which have not yet been saved.

## Supported Drag and Drop

Drag and drop is supported between all fields that contain or show Snomed Concepts within the editor.

When a drag operation begins, all fields that support accepting the drop operation will be highlighted in Green, as shown below.

Figure



When the concept is dropped on a concept editor field, the concept will simply be placed into the editor field.

However, if the concept is dropped on a highlighted label such as “Discernible Expression” then a context menu will appear asking how the dropped concept should be treated.

Figure

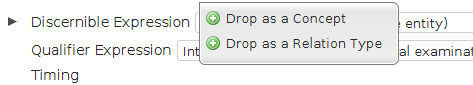


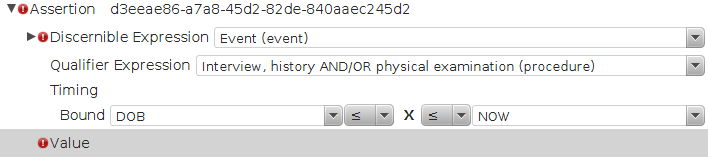
Figure 35 shows the menu displayed when a concept is dropped on “Discernible Expression”.

## Error Display

As shown above in Concept Editing, each Concept Editor will display an error indicator if the entered concept cannot be resolved.

Additionally, the Editor Tree will also show error indicators on the left of the nodes as necessary.

Figure



In the example above, an error indicator is being shown on “Value” because no Value is present – and Value is required in every LEGO. An error is being shown on “Discernible Expression” because the concept that was placed in the Discernible which is not a child of the concept Observable. An error is being shown on the Assertion node because it contains child node(s) with errors.

Hovering the mouse over any of the error indicators will give a detailed explanation of why it appears.

## Pending Concepts

Since the requires that all Discernible concepts be children of the Observable entity concept – many new concepts will need to be added to Snomed. In order to allow LEGO Creation to continue before the necessary concepts have been added, the Pending Concepts paradigm has been created within the Lego Editor.

Pending Concepts can be created with the “Add Pending Concept” option in the Edit Menu. These concepts will then appear in the Pending Concepts panel. Pending Concepts can be used anywhere in the editor as if they were a real Snomed Concept.

Any Pending Concepts that are created for editing purposes should be submitted back to the IHTSDO so that they can be properly added to the Snomed distribution.

When Pending Concepts are created within the Editor, they are stored in a tab-separated-value file - <Installation Path>/LegoEditor/pendingConcepts.tsv

While the editor is not running, this file can be edited manually if the user prefers for bulk concept addition. The format of the file is explained within the file itself.

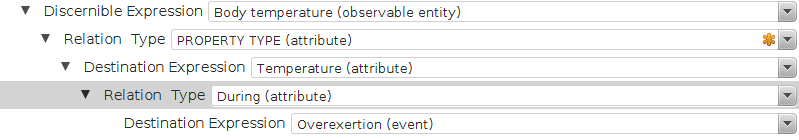
## Advanced Editing

The Lego data model allows a number of complex ways of combining Snomed concepts during the creation of a LEGO. Some examples follow.

### Infinite Nesting

Snomed concept Expressions and Relationships can be nested infinitely deep.

Figure

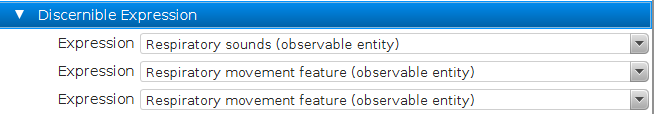


### Conjunctions

Discernible Expressions, Qualifier Expressions, Value Expressions and Destination Expressions each can be a single Snomed Concept, or, they can be a combination of an infinite number of concepts.

The Context menu on each of these Expression types allows the addition of more concepts. When another concept is added, the Editor will display the concepts as shown below.

Figure



All three of the Expressions listed in Figure 38 are a member of the concept conjunction that makes up the Discernible Expression.

The Discernible Expression Conjunction can have Relationships added to it, and each individual concept that is a member of the conjunction can also have Relationships added to them.

### Relationship Groups

In all places where Relationships are allowed, Relationship Groups are also allowed.

Figure

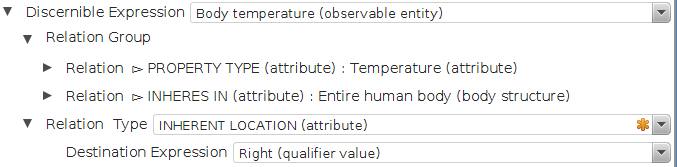


Figure 39 shows a Discernible Expression that contains a single Relationship that is not grouped, and two Relationships that are grouped.

### Measurements

When measurements are entered into the Editor, they can be represented as a Point, an Interval, or a Bound. The Measurement also allows an optional “Units” field.

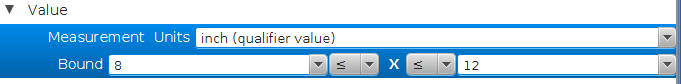
A Point is a single entry.

Figure

image40.png

A Bound is two entries – although only one of them is required.

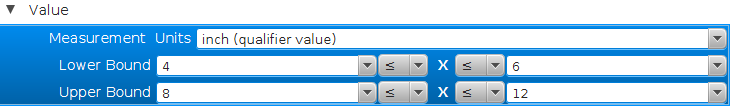
Figure



You may also specify whether the lower and upper limits are inclusive or exclusive.

An Interval is four entries – although only two of them are required. An interval is used to represent uncertainty in the endpoints of the bound.

Figure



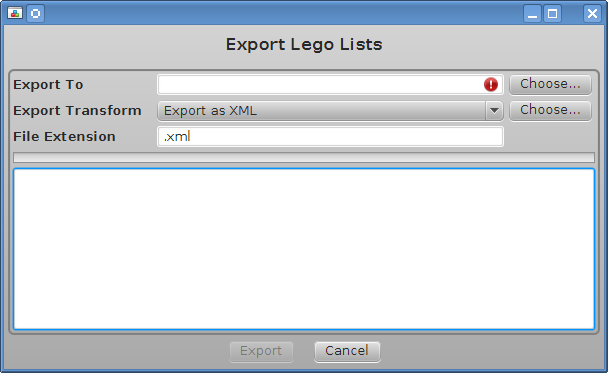
As above in the Bound, you can specify whether each value is inclusive or exclusive.

# Lego Exporting Options

The Lego Lists (and contained LEGOs) that are created by the Lego Editor can be exported into the native LEGO XML format, or they can be transformed into a number of other formats.

## Exporting

The Export window is launched by the File Menu Export option, or via a Context Menus in the Lego Lists Panel.

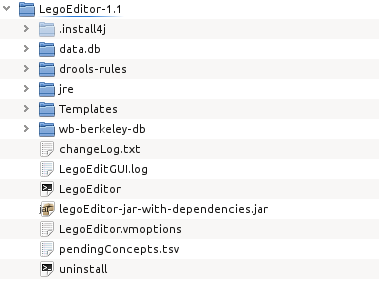


The Export Dialog requires a number of user entries.

* Export To – The folder to export the content to. Select by clicking the corresponding “Choose…” button.
* Export Transform – Select the Format of the resulting exported files.
  + Export as XML – Export Lego native XML format, which adheres to the Appendix B: Lego XML Schema
  + Export as XHTML – Export the Lego as a html file, specifically a XHTML 1.1 formatted html file
  + Export as browser viewable XML – Similar to the XML export, but wraps all of the XML tags in appropriate HTML tags so that the XML can be viewed in a web browser.
  + “Choose…” button – This button allows you to specify any valid XSL transform to apply to the Lego XML during export
* File Extension – The extension to use for the created files. Suggested values are populated based on the Export Transform selection.

Appendix A: Installed Files Overview

The following files make up a Lego Editor installation.



* .install4j – Files created by the installer to aid in upgrades and uninstall
* data.db – The database that stores the LEGOs that have been loaded into the editor. Deleting this folder will erase all Lego Lists from the editor.
* drools-rules – Content based rules that are applied to the Snomed concepts during editing. Drools files and be added or removed from this folder to enable or disable different rules.
* jre – The Java Runtime that the Lego Editor uses
* Templates – XML files that represent Templates that have been created by the user. Template XML files can be manually added, removed, or shared with others as desired.
* wb-berkeley-db – The Snomed database that is used within the Lego Editor.
* changeLog.txt – Developer notes about changes to the editor from version to version
* LegoEditGUI.log – Logfile of editor operations. Should be included with bug reports.
* LegoEditor – Platform specific program launcher
* lego-Editor-jar-with-dependencies.jar – Program libraries
* LegoEditor.vmoptions – The Java memory settings to use when launching the Lego Editor on your system. If the editor does not launch on your system, you may need to edit this file and lower the memory settings.
* pendingConcepts.tsv – The storage file for Pending Concepts
* uninstall – The uninstall program for the Lego Editor.

# Appendix B: Lego XML Schema

The XML Schema model that defines a Lego XML file is posted on the Collabnet project page for the Veterans Administration Project under the File Releases section.

<https://csfe.aceworkspace.net/sf/frs/do/viewRelease/projects.veterans_administration_project/frs.lego_editor.lego_schema_model>

The current version of the schema is 2013/03/06 – which is posted here:

<https://csfe.aceworkspace.net/sf/frs/do/downloadFile/projects.veterans_administration_project/frs.lego_editor.lego_schema_model/frs4820?dl=1>