# LexEVS 6.x OWL Export Guide

**Contents of this Page**

LexEVS 6.x provides two methods to export loaded ontologies to an OWL/RDF format. One is to use the LexEVS administration GUI; and the other is to use an export script on the command line.

## Preliminary Considerations

- The OWL/RDF exporter is based on Jena 2.6.3. While it exports an OWL file, it also builds up a triple store. The triple store tables stored in the same same database that LexEVS uses.
- The LexEVS 6.0 retrieval API has the limitation of retrieving the association that is from an entity to a data/value. Thus, the OWL/RDF exporter, based on LexEVS retrieval API, has the limitation of handling the owl:hasValue, owl:maxCardinality, owl:minCardinality, owl:cardinality constraints as well.
- The OWL/RDF exporter now can convert OWL, RDF, OBO, and UML formats to OWL/RDF. If the terminology format is not on this list, the exported OWL ontology may be incorrect.

> Note that OWL2 is not currently supported, nor is an export to the OWL2 format.  Much of what is loaded to LexEVS is currently OWL2, and cannot be exported.  See JIRA Item LEXEVS-3661

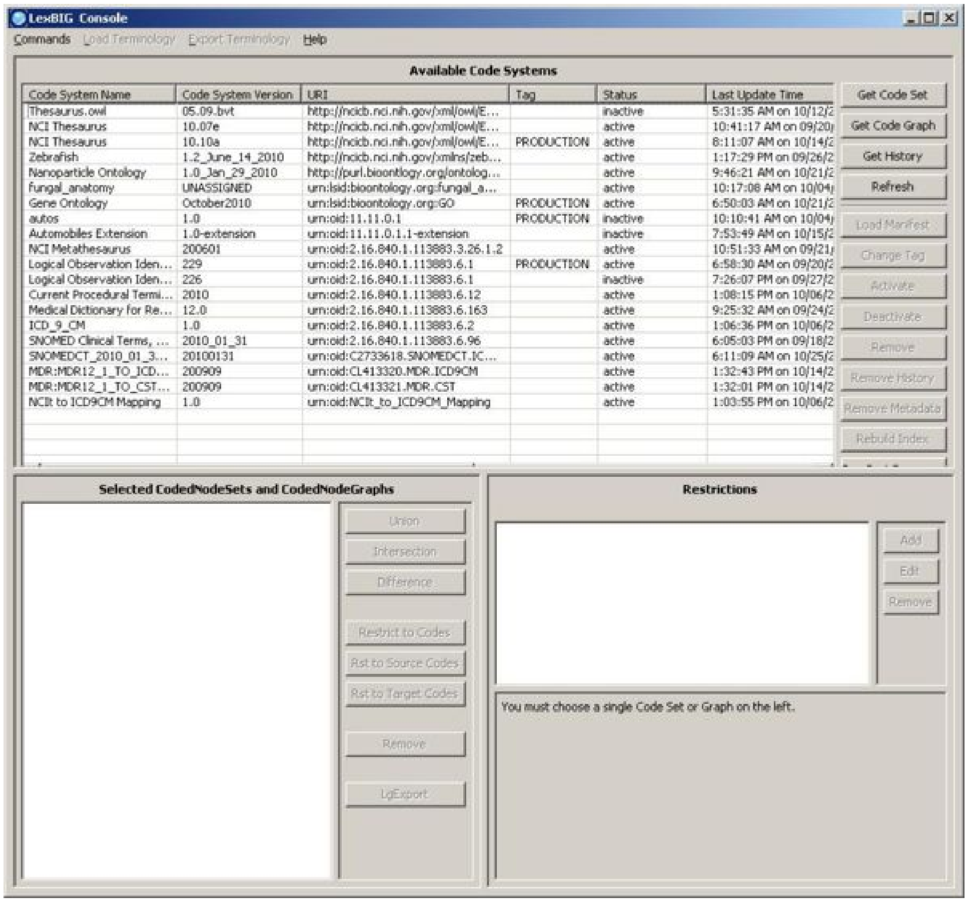
## Running an OWL/RDF Export with the Administration GUI

If you chose to install the LexEVS GUI when you installed LexEVS, you will have a 'gui' folder inside of your LexEVS home directory. Assuming you installed the GUI for all operating systems, you will have the following programs under the 'gui' folder:

- Linux_64-lbGUI.sh
- Linux-lbGUI.sh
- OSX-lbGUI.command
- Windows-lbGUI.bat

| Step | Action |
|---|---|

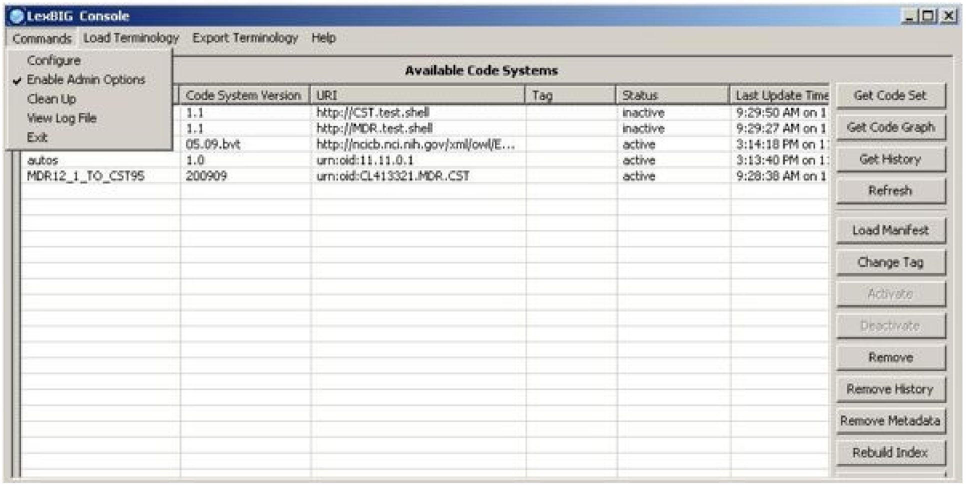
1. Launch the GUI by executing the appropriate executable for your platform. You will be presented with an application that looks like this. If the Available Code System table is empty, you will need to load at least one ontology.



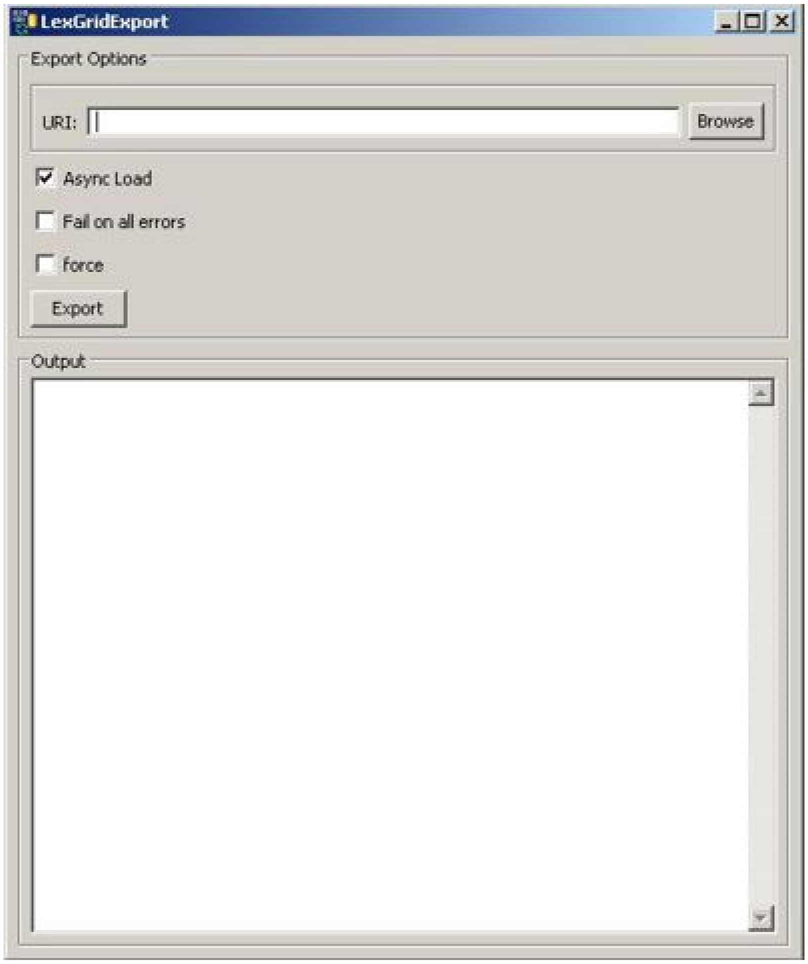
| Code System Name | Code System Version | URI | Tag | Status | Last Update Time |
|---|---|---|---|---|---|
| Thesaurus.owl | 05.09.bvt | http://ncicb.nci.nih.gov/xml/owl/E... | | inactive | 5:31:35 AM on 10/12/2 |
| NCI Thesaurus | 10.07e | http://ncicb.nci.nih.gov/xml/owl/E... | | active | 10:41:17 AM on 09/20/ |
| NCI Thesaurus | 10.10a | http://ncicb.nci.nih.gov/xml/owl/E... | PRODUCTION | active | 8:11:07 AM on 10/14/2 |
| Zebrafish | 1.2_June_14_2010 | http://ncicb.nci.nih.gov/xmlns/zeb... | | active | 1:17:29 PM on 09/26/2 |
| Nanoparticle Ontology | 1.0_Jan_29_2010 | http://purl.bioontlogy.org/ontolog... | | active | 9:46:21 AM on 10/21/2 |
| fungal_anatomy | UNASSIGNED | urn:lsid:bioontology.org:fungal_a... | | active | 10:17:08 AM on 10/04/ |
| Gene Ontology | October2010 | urn:lsid:bioontology.org:GO | PRODUCTION | active | 6:50:03 AM on 10/21/2 |
| autos | 1.0 | urn:oid:11.11.0.1 | PRODUCTION | inactive | 10:10:41 AM on 10/04/ |
| Automobiles Extension | 1.0-extension | urn:oid:11.11.0.1.1-extension | | inactive | 7:53:49 AM on 10/15/2 |
| NCI Metathesaurus | 200601 | urn:oid:2.16.840.1.113883.3.26.1.2 | | active | 10:51:33 AM on 09/21/ |
| Logical Observation Iden... | 229 | urn:oid:2.16.840.1.113883.6.1 | PRODUCTION | active | 6:58:30 AM on 09/20/2 |
| Logical Observation Iden... | 226 | urn:oid:2.16.840.1.113883.6.1 | | inactive | 7:26:07 PM on 09/27/2 |
| Current Procedural Termi... | 2010 | urn:oid:2.16.840.1.113883.6.12 | | active | 1:08:15 PM on 10/06/2 |
| Medical Dictionary for Re... | 12.0 | urn:oid:2.16.840.1.113883.6.163 | | active | 9:25:32 AM on 09/24/2 |
| ICD_9_CM | 1.0 | urn:oid:2.16.840.1.113883.6.2 | | active | 1:06:36 PM on 10/06/2 |
| SNOMED Clinical Terms, ... | 2010_01_31 | urn:oid:2.16.840.1.113883.6.96 | | active | 6:05:03 PM on 09/18/2 |
| SNOMEDCT_2010_01_3... | 20100131 | urn:oid:C2733618.SNOMEDCT.IC... | | active | 6:11:09 AM on 10/25/2 |
| MDR:MDR12_1_TO_ICD... | 200909 | urn:oid:CL413320.MDR.ICD9CM | | active | 1:32:43 PM on 10/14/2 |
| MDR:MDR12_1_TO_CST... | 200909 | urn:oid:CL413321.MDR.CST | | active | 1:32:01 PM on 10/14/2 |
| NCIt to ICD9CM Mapping | 1.0 | urn:oid:NCIt_to_ICD9CM_Mapping | | active | 1:03:55 PM on 10/06/2 |

You must choose a single Code Set or Graph on the left.

2. Selecting 'Enable Admin Options' provides administrative access to all commands.



| | Code System Version | URI | Tag | Status | Last Update Time |
|---|---|---|---|---|---|
| | 1.1 | http://CST.test.shell | | inactive | 9:29:50 AM on 1 |
| | 1.1 | http://MDR.test.shell | | inactive | 9:29:27 AM on 1 |
| | 05.09.bvt | http://ncicb.nci.nih.gov/xml/owl/E... | | active | 3:14:18 PM on 1 |
| autos | 1.0 | urn:oid:11.11.0.1 | | active | 3:13:40 PM on 1 |
| MDR12_1_TO_CST95 | 200909 | urn:oid:CL413321.MDR.CST | | active | 9:28:38 AM on 1 |

| | |
|---|---|
| 3. Select an active terminology (if the code system status is 'inactive', you can use the Activate button to activate it).<br>Click on 'Export as OWL/RDF' button on the 'Export Terminology' menu, the exporter window will pop up.<br>Fill in the parameters. See the export parameter descriptions in the Administration Guide.<br>Click the 'Export' button, the exporter will generate the OWL file to the specified path. | <br> LexGridExport<br>Export Options<br>URI:<br>Browse<br>Export<br>Output<br>☑ Async Load<br>□ Fail on all errors<br>□ force |

## Running an OWL/RDF Export Script from the Command Line

After installing LexEVS, you will have an 'admin' folder inside of your LexEVS base installation. The OWL/RDF exporter shell script is in the following folder (for Linux or Windows):

```
ExportOwlRdf.[sh|bat]
```

This command will export OWL/RDF without a GUI. For detailed information please refer to the command line parameters for this command in the *Administration Guide*.

Examples:

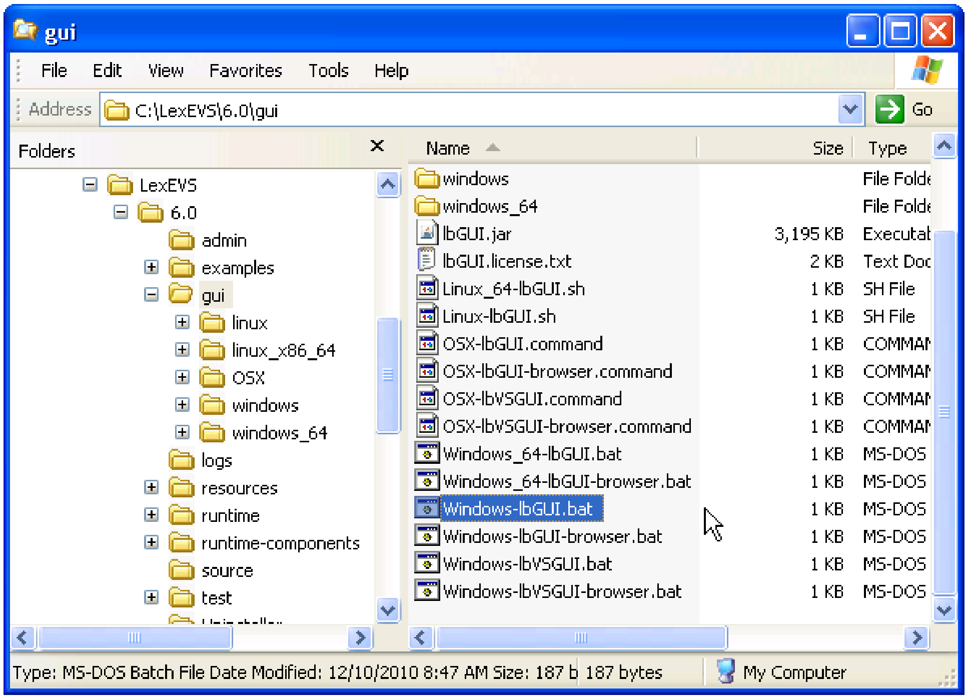
```
 ExportOwlRdf -out "file:///path/to/dir" -f
 ExportOwlRdf -out "file:///path/to/dir" -u "sample" -v "1.0" -f
```

## Administration GUI export walk-through example

This walk through will show you how to load a terminology and then export it in OWL/RDF format.

| Step | Action |
|---|---|

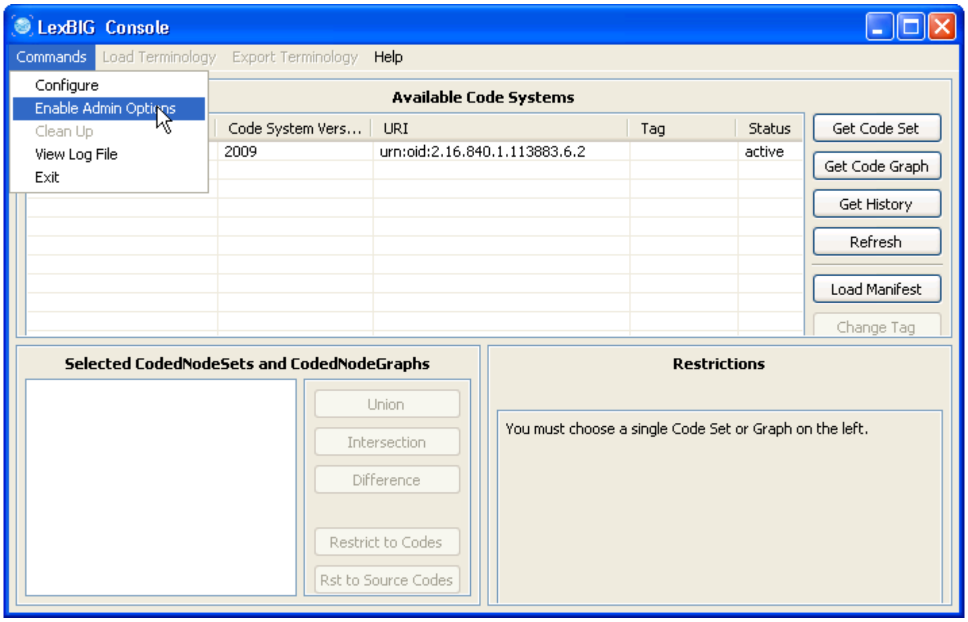
1. Launch the administration GUI.



| Name | Size | Type |
|---|---|---|
| windows | | File Folde |
| windows_64 | | File Folde |
| lbGUI.jar | 3,195 KB | Executal |
| lbGUI.license.txt | 2 KB | Text Doc |
| Linux_64-lbGUI.sh | 1 KB | SH File |
| Linux-lbGUI.sh | 1 KB | SH File |
| OSX-lbGUI.command | 1 KB | COMMAN |
| OSX-lbGUI-browser.command | 1 KB | COMMAN |
| OSX-lbVSGUI.command | 1 KB | COMMAN |
| OSX-lbVSGUI-browser.command | 1 KB | COMMAN |
| Windows_64-lbGUI.bat | 1 KB | MS-DOS |
| Windows_64-lbGUI-browser.bat | 1 KB | MS-DOS |
| Windows-lbGUI.bat | 1 KB | MS-DOS |
| Windows-lbGUI-browser.bat | 1 KB | MS-DOS |
| Windows-lbVSGUI.bat | 1 KB | MS-DOS |
| Windows-lbVSGUI-browser.bat | 1 KB | MS-DOS |

Type: MS-DOS Batch File Date Modified: 12/10/2010 8:47 AM Size: 187 b 187 bytes My Computer

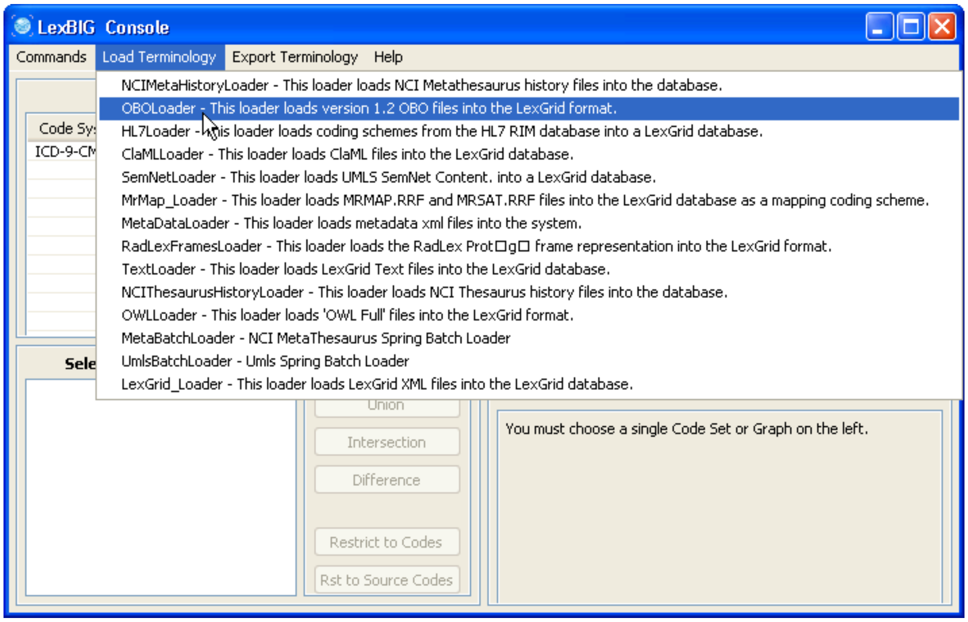
2. Enable the administration option.



| Code System Vers... | URI | Tag | Status |
|---|---|---|---|
| 2009 | urn:oid:2.16.840.1.113883.6.2 | | active |

You must choose a single Code Set or Graph on the left.

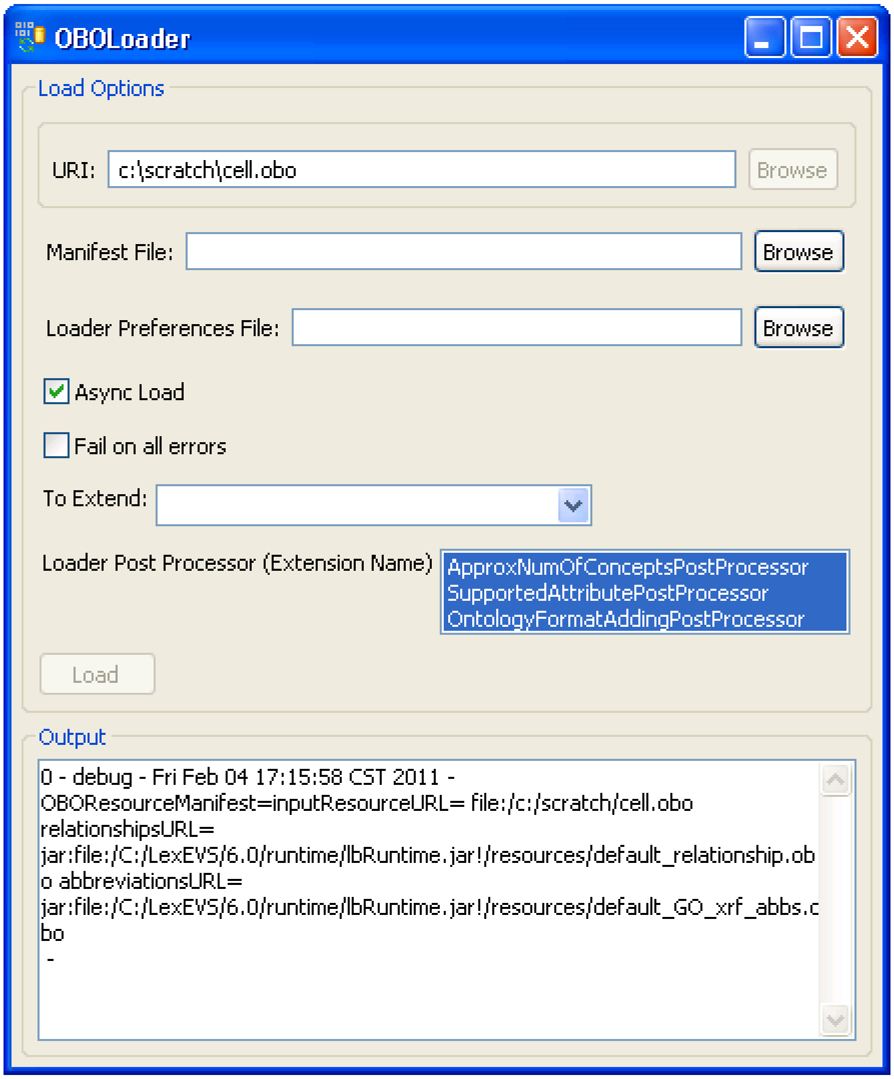
3. In this example, we use the cell.obo ontology as the example ontology, so we choose OBOloader.



NCIMetaHistoryLoader - This loader loads NCI Metathesaurus history files into the database.
OBOLoader - This loader loads version 1.2 OBO files into the LexGrid format.
HL7Loader - This loader loads coding schemes from the HL7 RIM database into a LexGrid database.
ClaMLLoader - This loader loads ClaML files into the LexGrid database.
SemNetLoader - This loader loads UMLS SemNet Content. into a LexGrid database.
MrMap_Loader - This loader loads MRMAP.RRF and MRSAT.RRF files into the LexGrid database as a mapping coding scheme.
MetaDataLoader - This loader loads metadata xml files into the system.
RadLexFramesLoader - This loader loads the RadLex Prot□g□ frame representation into the LexGrid format.
TextLoader - This loader loads LexGrid Text files into the LexGrid database.
NCIThesaurusHistoryLoader - This loader loads NCI Thesaurus history files into the database.
OWLLoader - This loader loads 'OWL Full' files into the LexGrid format.
MetaBatchLoader - NCI MetaThesaurus Spring Batch Loader
UmlsBatchLoader - Umls Spring Batch Loader
LexGrid_Loader - This loader loads LexGrid XML files into the LexGrid database.

You must choose a single Code Set or Graph on the left.

4. Provide cell.obo file's path. If you don't have a manifest file or preferences file then leave these fields blank. Click on the 'Load' button. After a few minutes, the loading process is done. You can find 'End State: complete' in the Output message box. Close this window, and return to the administration GUI, a 'cell' record should show up on the Available Code Systems list.



URI: c:\scratch\cell.obo

Manifest File:

Loader Preferences File:

☑ Async Load

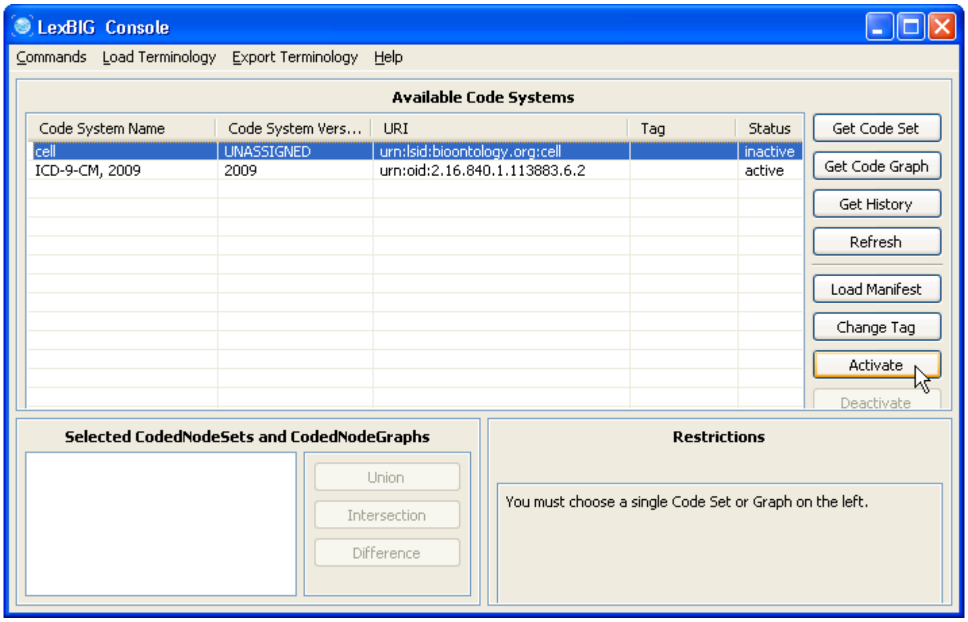
□ Fail on all errors

To Extend:

Loader Post Processor (Extension Name): ApproxNumOfConceptsPostProcessor, SupportedAttributePostProcessor, OntologyFormatAddingPostProcessor

```
0 - debug - Fri Feb 04 17:15:58 CST 2011 -
OBOResourceManifest=inputResourceURL= file:/c:/scratch/cell.obo
relationshipsURL=
jar:file:/C:/LexEVS/6.0/runtime/lbRuntime.jar!/resources/default_relationship.ob
o abbreviationsURL=
jar:file:/C:/LexEVS/6.0/runtime/lbRuntime.jar!/resources/default_GO_xrf_abbs.o
bo
 -
```

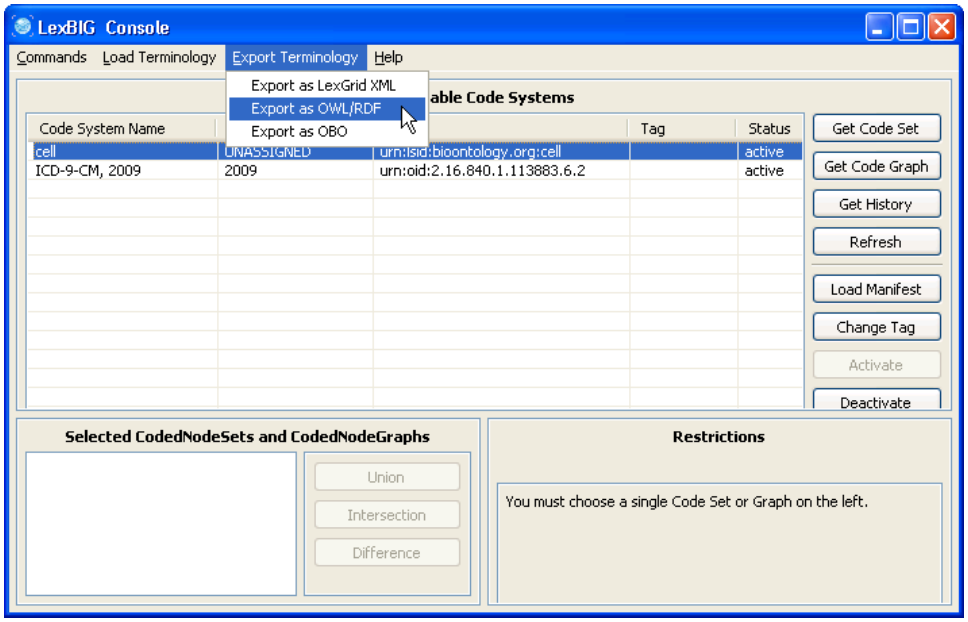
5. Select the cell ontology row and click the 'Activate' button. The status column of 'cell' should change to active.



| Code System Name | Code System Vers... | URI | Tag | Status |
|---|---|---|---|---|
| cell | UNASSIGNED | urn:lsid:bioontology.org:cell | | inactive |
| ICD-9-CM, 2009 | 2009 | urn:oid:2.16.840.1.113883.6.2 | | active |

You must choose a single Code Set or Graph on the left.

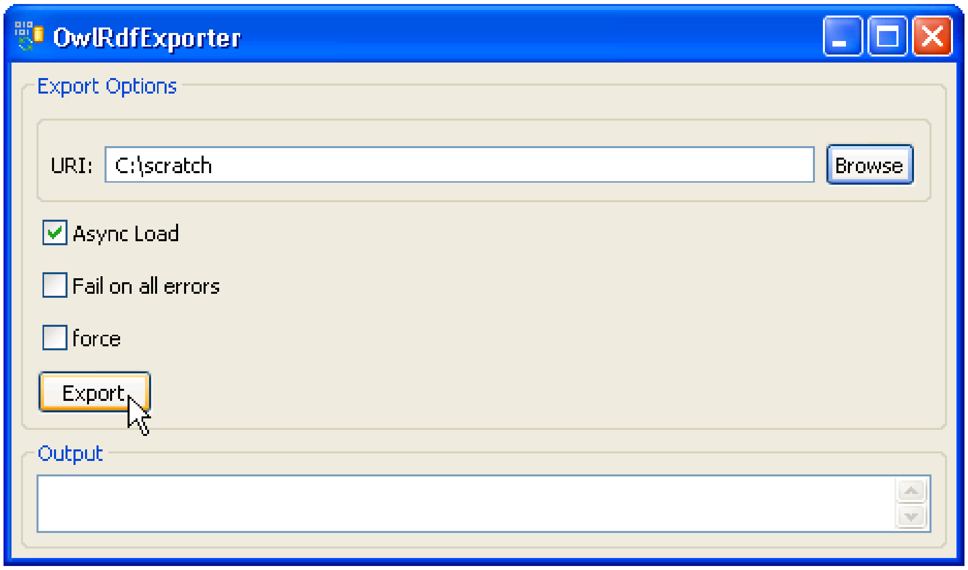
6. Make sure the cell ontology is still selected. Go to the 'Export Terminology' menu, click on 'Export as OWL/RDF' in the drop down menu.



| Code System Name | | | Tag | Status |
|---|---|---|---|---|
| cell | UNASSIGNED | urn:lsid:bioontology.org:cell | | active |
| ICD-9-CM, 2009 | 2009 | urn:oid:2.16.840.1.113883.6.2 | | active |

You must choose a single Code Set or Graph on the left.

7. Provide a path in the 'URI' text box. The exported file name is automatically generated by the system, so we don't need to specify the file name. Click on the 'Export' button.



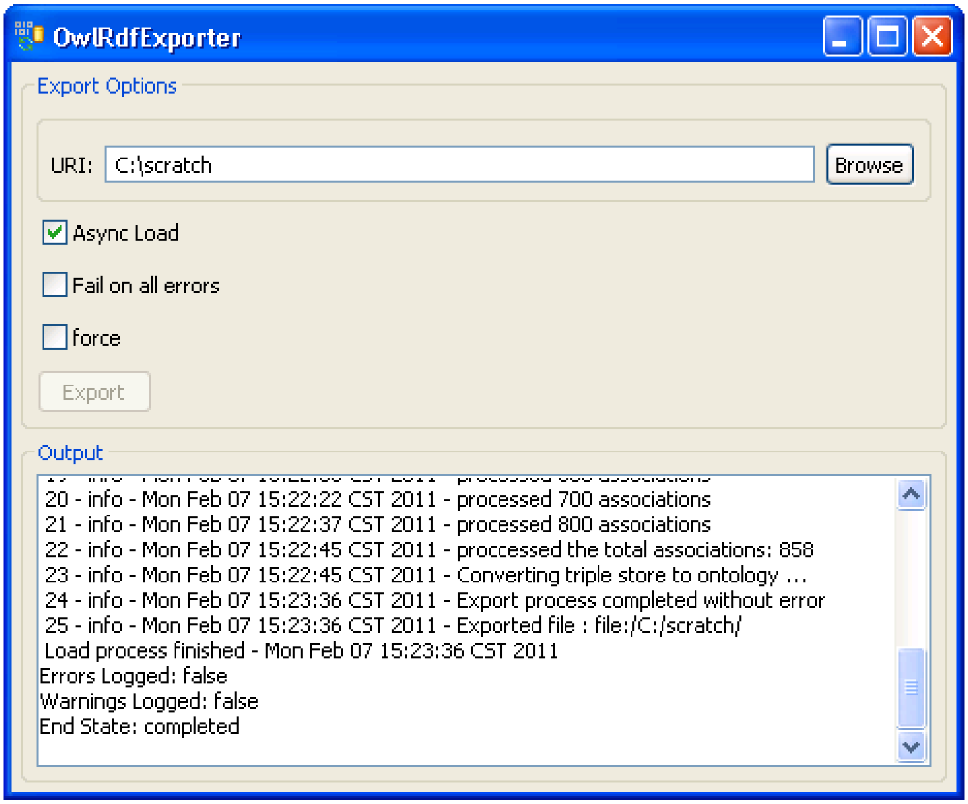
URI: C:\scratch

☑ Async Load

□ Fail on all errors

□ force

8. A set of status messages will show up in the Output text box. When the 'End State: completed' statement pops up, the exporting process is complete.



URI: C:\scratch

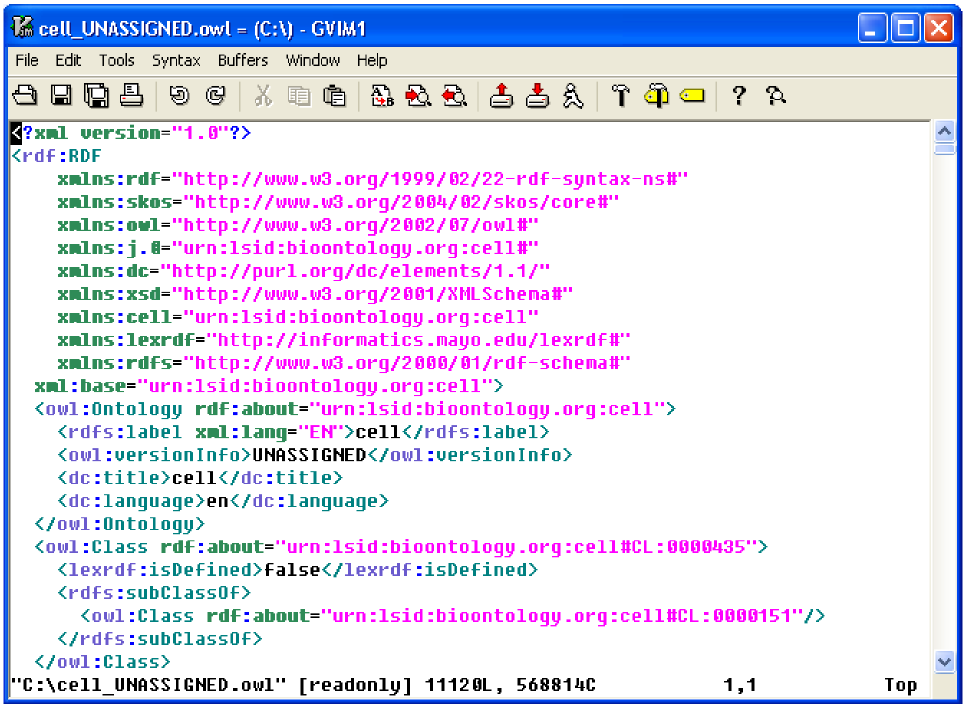
☑ Async Load

□ Fail on all errors

□ force

```
 20 - info - Mon Feb 07 15:22:22 CST 2011 - processed 700 associations
 21 - info - Mon Feb 07 15:22:37 CST 2011 - processed 800 associations
 22 - info - Mon Feb 07 15:22:45 CST 2011 - proccessed the total associations: 858
 23 - info - Mon Feb 07 15:22:45 CST 2011 - Converting triple store to ontology ...
 24 - info - Mon Feb 07 15:23:36 CST 2011 - Export process completed without error
 25 - info - Mon Feb 07 15:23:36 CST 2011 - Exported file : file:/C:/scratch/
 Load process finished - Mon Feb 07 15:23:36 CST 2011
Errors Logged: false
Warnings Logged: false
End State: completed
```

9. The convention of the exported file name is: <coding system name>_<version>.owl. In this example, cell ontology has no version information, so 'UNASSIGNED' is used as the version. The name of the OWL file in this case will be cell_UNASSIGNED.owl and it is in the C drive's root directory.
You can check the exported OWL file with a text editor. Yours may not show highlighting for the XML syntax.


cell_UNASSIGNED.owl = (C:\) - GVIM1
File Edit Tools Syntax Buffers Window Help

```
<?xml version="1.0"?>
<rdf:RDF
    xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
    xmlns:skos="http://www.w3.org/2004/02/skos/core#"
    xmlns:owl="http://www.w3.org/2002/07/owl#"
    xmlns:j.0="urn:lsid:bioontology.org:cell#"
    xmlns:dc="http://purl.org/dc/elements/1.1/"
    xmlns:xsd="http://www.w3.org/2001/XMLSchema#"
    xmlns:cell="urn:lsid:bioontology.org:cell"
    xmlns:lexrdf="http://informatics.mayo.edu/lexrdf#"
    xmlns:rdfs="http://www.w3.org/2000/01/rdf-schema#"
  xml:base="urn:lsid:bioontology.org:cell">
  <owl:Ontology rdf:about="urn:lsid:bioontology.org:cell">
    <rdfs:label xml:lang="EN">cell</rdfs:label>
    <owl:versionInfo>UNASSIGNED</owl:versionInfo>
    <dc:title>cell</dc:title>
    <dc:language>en</dc:language>
  </owl:Ontology>
  <owl:Class rdf:about="urn:lsid:bioontology.org:cell#CL:0000435">
    <lexrdf:isDefined>false</lexrdf:isDefined>
    <rdfs:subClassOf>
      <owl:Class rdf:about="urn:lsid:bioontology.org:cell#CL:0000151"/>
    </rdfs:subClassOf>
  </owl:Class>
"C:\cell_UNASSIGNED.owl" [readonly] 11120L, 568814C           1,1           Top
```

## Command Line Script Exporting Example

We still use the cell.obo as our example ontology and assume that it has been loaded already. It is straightforward to run a command like this instead of using the GUI:

```
ExportOwlRdf -out "file:///home/MyHome" -u "cell" -v "1.0" -f
```

A set of statements will show up, like the following, if it runs successfully:

```
Output from command line execution...
```

A file named "cell_UNASSIGNED.owl" will be generated and saved under the specified path. The content of this file is the same as the file generated by the administration GUI.