National Cancer Institute

U.S. National Institutes of Health | www.cancer.gov

caBIG Knowledge Center
A part of the Enterprise Support Network

- Home
- Knowledge Centers
  - caGrid
  - Clinical Trials Management Systems
  - Data Sharing and Intellectual Capital
  - Molecular Analysis Tools
  - Tissue/Biospecimen Banking and Technology Tool
  - Vocabulary
- Discussion Forums
  - caGrid
  - Clinical Trials Management Systems
  - Data Sharing and Intellectual Capital
  - Molecular Analysis Tools
  - Tissue/Biospecimen Banking and Technology Tool
  - Vocabulary
- Bugs/Feature Requests
- Development Code Repository

LexEVS 5.0 Administration Guide

From Vocab_Wiki

LexEVS 5.0 Administration Guide > LexBig and LexEVS > LexEVS Version 5.0 > LexEVS 5.0 Documentation > LexEVS 5.0 Administration Guide

Contents

LexEVS Model/DB (back-end) Administration

- Environment configuration from the perspective of an existing installation

LexEVS Configuration Options

The LexEVS software, documentation, indexes, and system logs are located in the {LEXBIG_DIRECTORY} (e.g. /usr/local/packages/LexBIG or c:\lexbig). These files may be part of the local file system and may require backup procedures to meet servicability and recovery requirements for your organization.

LexEVS uses basic database indexes, but also includes a separate indexing facility using Apache Lucene. Lucene Index files are stored in a directory as specified in the `config.props` file `index_location` variable.

What is Coding Scheme Manifest?

A “Coding Scheme Manifest” (or simply “manifest” as used interchangeably in this document) allows the user to set values for a coding scheme while loading or converting a LexGrid XML", "NCI MetaThesaurus", "NCI OWL","OWL", "OBO", "UMLS RRF File", or"HL7 RIM Database" source to LexGrid format.

What is Coding Scheme?

Coding Scheme is the term that is used to represent an ontology/terminology being loaded or converted. In the LexGrid data model a terminology is represented as a coding scheme and it can reference other coding schemes. An example of coding scheme is “Amino Acid” which is described in the “amino acid.owl” file.

A Coding Scheme has some meta information about it; values like ‘formal name’, ‘local names’, ‘default language’, ‘version’, ‘copyright’, ‘sources’ to name some.

Why do we need a Coding Scheme Manifest?

When a terminology is being converted to the LexGrid data model from its native format (in this case OWL), Coding Scheme information is read from the source file. Sometimes values may be missing (not provided or invalid) or the author/user of the terminology wants to override or set default values despite (or in addition to) what is provided in the source file. This can be accomplished using “manifest” files along with the source file.

How do we create a Coding Scheme Manifest file?

A coding scheme manifest file is a valid XML file, conforming to the schema defined by http://LexGrid.org/schema/LexBIG/2007/01/CodingSchemeManifestList.xsd. This XML file can define values for one or more coding schemes you are dealing with. Some coding scheme meta-information may not easily map to information in the source file. In this case a manifest file is of great help to bridge the gap and control the information flow while mapping to the LexGrid model. A detailed model of the LexGrid Coding Scheme and its fields can be found online [1]. Structure of the schema for the manifest file is explained in the following table (manifest components refer to the original LexGrid model schema namespaces and types):

- Coding Scheme Manifest entry field: **id**
  - Type: lgCommon:registeredName
  - Required: Yes
  - Override flag set: Not applicable
  - Description:

The registered name is the key used to find a coding scheme (for example a unique URL or namespace by which other people access same coding scheme). This String value will be used to identify the manifest entry in the manifest file for the coding scheme too. For example the registered name for coding scheme “Amino-acid” is http://www.co-ode.org/ontologies/amino-acid/2006/05/18/amino-acid.owl#. This string is also set as the coding scheme’s registered name field in the LexGrid model.

- Coding Scheme Manifest entry field: **codingScheme**
  - Type: lgBuiltin:localId
  - Required: No
  - Override flag set: Yes
  - Description:

This value will be set for ‘coding scheme name’ in the LexGrid format counterpart. If the override flag is set to ‘true’, the value provided in the source file will be replaced with this one. Otherwise, this value is treated as a default value and used only if the value is not provided in the source file.

- Coding Scheme Manifest entry field: **entityDescription**
  - Type: lgCommon:entityDescription
  - Required: No
  - Override flag set: Yes
  - Description:

This value will be set for ‘coding scheme description’ in the LexGrid format counterpart. If the override flag is set to ‘true’, the value provided in the source file will be replaced with this one. Otherwise, this value is treated as a default value and used only if the value is not provided in the source file.

- Coding Scheme Manifest entry field: **formalName**
  - Type: lgBuiltin:tsCaseIgnoreIA5String
  - Required: No
  - Override flag set: Yes
  - Description:

This value will be set for ‘coding scheme formal name’ in the LexGrid format counterpart. If the override flag is set to ‘true’, the value provided in the source file will be replaced with this one. Otherwise, this value is treated as a default value and used only if the value is not provided in the source file.

- Coding Scheme Manifest entry field: **registeredName**
  - Type: lgCommon:registeredName
  - Required: No
  - Override flag set: Yes
  - Description:

This value will be set for ‘coding scheme registered name’ in the LexGrid format counterpart. If the override flag is set to ‘true’, the value provided in the source file will be replaced with this one. Otherwise, this value is treated as a default value and used only if the value is not provided in the source file.

- Coding Scheme Manifest entry field: **defaultLanguage**
  - Type: lgCommon:defaultLanguage
  - Required: No
  - Override flag set: Yes
  - Description:

This value will be set for ‘coding scheme default language’ in the LexGrid format counterpart. If the override flag is set to ‘true’, the value provided in the source file will be replaced with this one. Otherwise, this value is treated as a default value and used only if the value is not provided in the source file.

- Coding Scheme Manifest entry field: **representsVersion**
  - Type: lgCommon:version
  - Required: No
  - Override flag set: Yes
- Description:

This value will be set for ‘coding scheme version’ in the LexGrid format counterpart. If the override flag is set to ‘true’, the value provided in the source file will be replaced with this one. Otherwise, this value is treated as a default value and used only if the value is not provided in the source file.

- Coding Scheme Manifest entry field: **localName**
  - Type: lgBuiltin:tsCaseIgnoreIA5String
  - Required: No
  - "To Add" flag set: Yes
  - Description:

This value will be added for ‘coding scheme local names’. If the add flag is set to ‘true’, this value will be added to the list of local names (if not there already). Otherwise, this value is treated as the default value and used only if the value is not provided in the source file.

- Coding Scheme Manifest entry field: **source**
  - Type: lgCommon:source
  - Required: No
  - "To Add" flag set: Yes
  - Description:

This value will be added for ‘coding scheme sources’. If the add flag is set to ‘true’, this value will be added to the list of sources (if not there already). Otherwise, this value is treated as the default value and used only if the value is not provided in the source file.

- Coding Scheme Manifest entry field: **copyright**
  - Type: lgCommon:text
  - Required: No
  - Override flag set: Yes
  - Description:

This value will be set for ‘coding scheme copyright’ in the LexGrid format counterpart. If the override flag is set to ‘true’, the value provided in the source file will be replaced with this one. Otherwise, this value is treated as a default value and used only if the value is not provided in the source file.

- Coding Scheme Manifest entry field: **mappings**
    - Type: lgCS:mappings
- Required: No
- "To Add" flag set: Yes
- Description:

This value will be added for ‘coding scheme mappings’. If the add flag is set to ‘true’, this value will be added to the list of mappings (if not there already). Otherwise, this value is treated as the default value and used only if the value is not provided in the source file.

- Coding Scheme Manifest entry field: **associationDefinitions**
    - Type: lgRel:association
    - Required: No
    - "To Add" flag set: Yes
    - Description:

This value will be added for ‘coding scheme associations’. If the add flag is set to ‘true’, this value will be added to the list of associations (if not there already). Otherwise, this value is treated as the default value and used only if the value is not provided in the source file.

(Note: This option is used internally by the system to provide default recognition of some common associations. It is typically not necessary to provide this value, however, since association definitions are automatically derived from the source.)

What code changes may be required to use a manifest file?

If you want to use the manifest file, you can supply the manifest file URI to the following methods when Loading NCI OWL or generic OWL Loads:

"org.LexGrid.LexBIG.Extensions.Load.OWL_Loader.load()"

"org.LexGrid.LexBIG.Extensions.Load.OWL_Loader.validate()"

An example code snipped:

```
LexBIGService lbs = new LexBIGServiceImpl();
   LexBIGServiceManager lbsm = lbs.getServiceManager(null);
   OWL_Loader loader = (OWL_Loader) lbsm.getLoader(“OWLLoader”);

if (toValidateOnly)
  {
       loader.validate(source, manifest, vl);
       System.out.println("VALIDATION SUCCESSFUL");
}
   else
  {
       loader.load(new File("resources/testData/amino-
  cid.owl").toURI(),
  new File("resources/testData/aa-manifest.xml").toURI(),true,
  true);
}
```

For all other manifest loads the following methods are employed.

```
// Find the registered extension handling this type of load
LexBIGService lbs = new
```

```
LexBIGServiceImpl();
```

```
LexBIGServiceManager lbsm = lbs.getServiceManager(null);
HL7_Loader loader = (HL7_Loader)lbsm.getLoader
```

```
(org.LexGrid.LexBIG.Impl.loaders
```

```
.HL7LoaderImpl.name);
// updated to include manifest
loader.setCodingSchemeManifestURI(manifest);
// updated to include loader
```

```
preferences
```

```
loader.setLoaderPreferences(loaderPrefs);
loader.load(dbPath, stopOnErrors, true);
```

Database Configuration

BEFORE YOU BEGIN	This section provides an overview of the components as related to system adminstration, backup, and recovery. Individual organizations may have there own backup and diaster recovery procedure.

Database systems as described in the section *Required Software—Not Included in LexEVS* provide the storage for vocabularies loaded into LexEVS. For each vocabulary version loaded into LexEVS a new database is created. As defined in the `config.props` files the `db_prefix` variable is used to create the database name.

For example with db_`prefix=lexbig`, each new vocabulary version that is loaded a new database is created using an incremental counter.

- lexbig1
- lexbig2
- lexbig3
- lexbigN

Depending on backup strategy, system administrators will need to be aware that multiple databases are being created and may need backup procedures to meet servicability and recovery requirements for your organization.

MySQL Configuration

MySQL Configuration Properties

PostGreSQL Configuration

PostgreSQL configuration properties

LexEVS Server Administration

Configuration Options

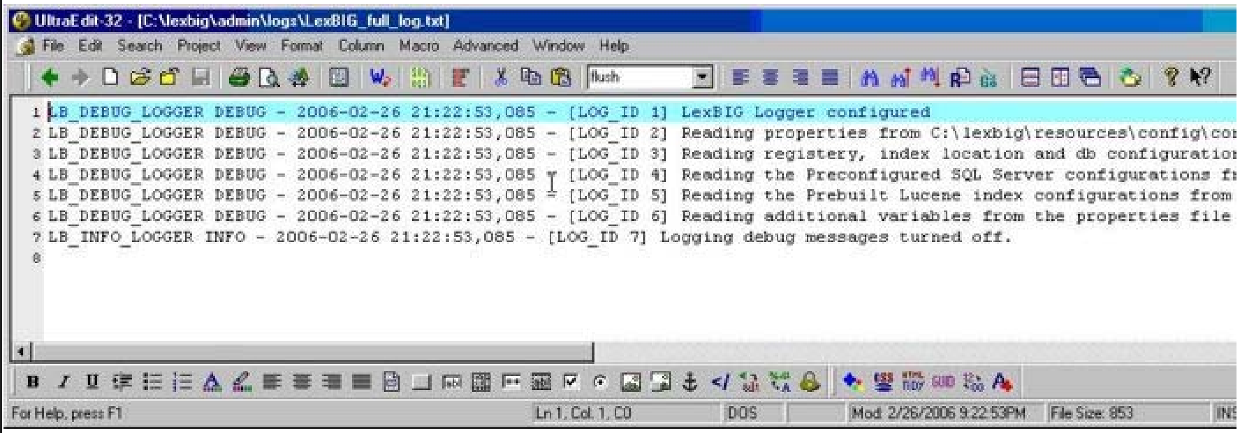
Server configuration properties

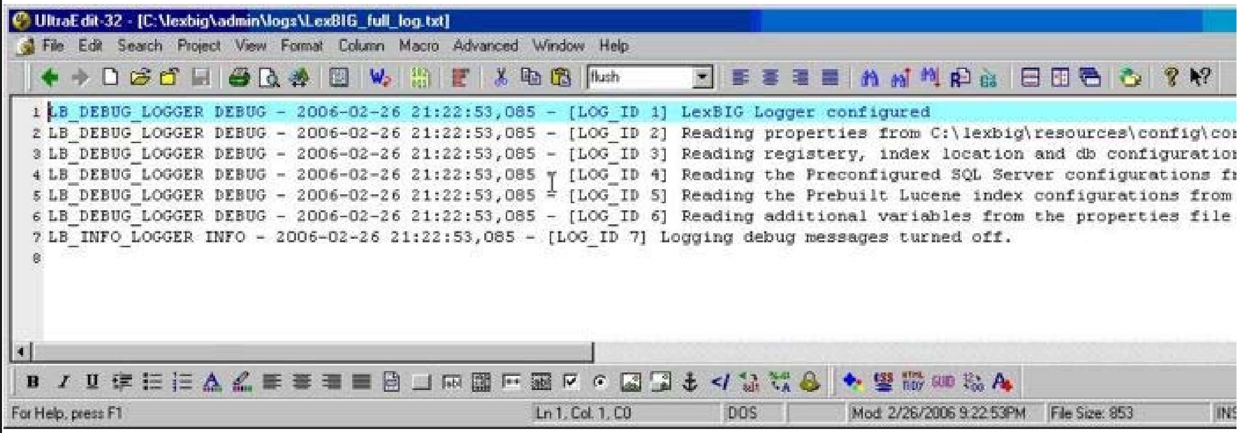
System Monitoring and Logging

This section describes the configuration and use of LexEVS logs for system monitoring and debugging. The LexEVS service uses a set of log files. The log files are stored based on the LexEVS lbconfig.props file settings. Refer to Modifying the config.props file for LexEVS on page 23 for additional detail configuration parameters.

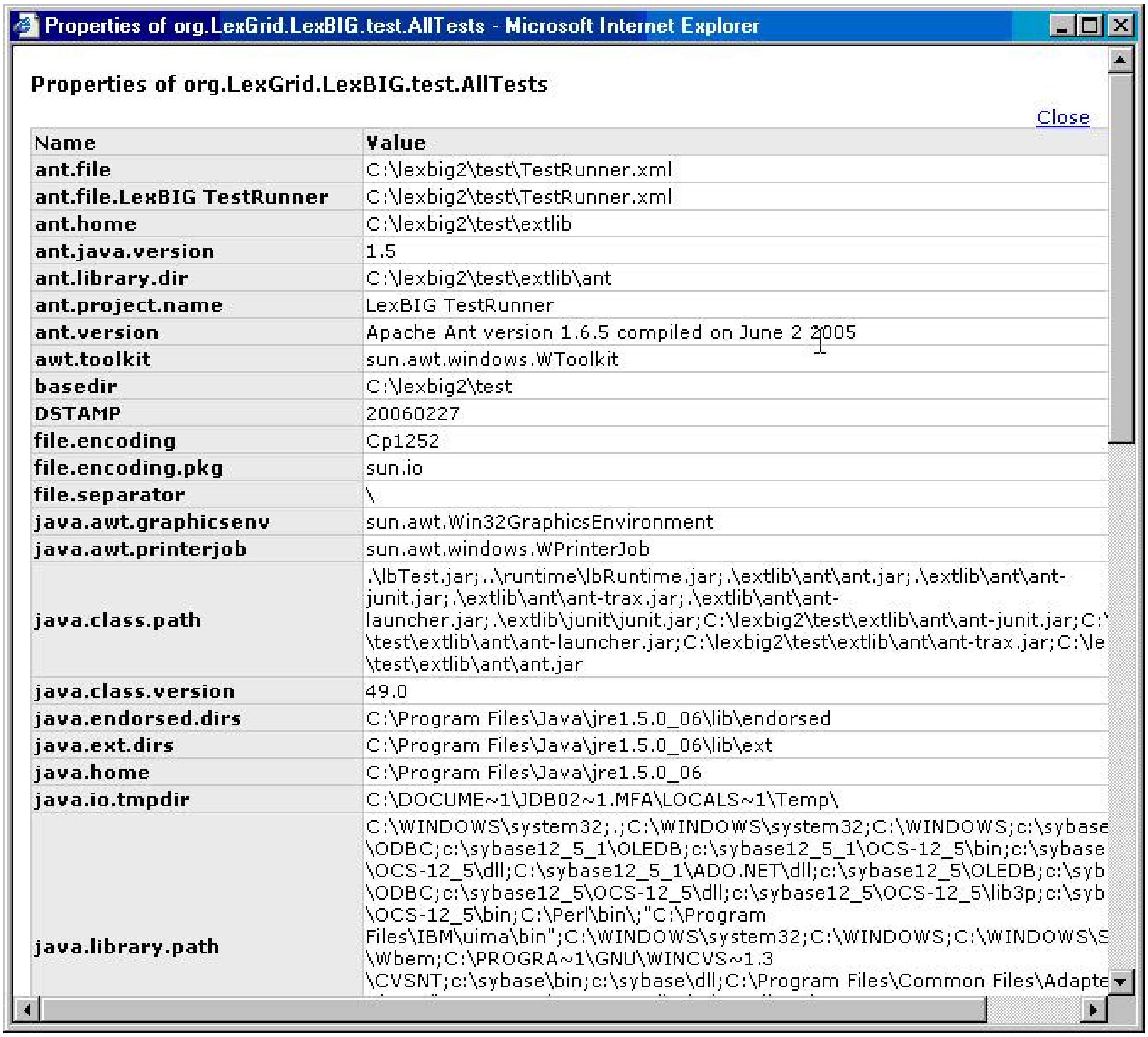
To view the LexEVS service and load log files perform the following steps.

Step	Action
1	Change directory to LexEVS administration directory based on the settings of the `config.props log_file_location` setting.

2	Open LexBIG_full_log.txt using text editor to review details about LexEVS service.
	
	UltraEdit-32 - [C:\lexbig\admin\logs\LexBIG_full_log.txt]
	`LB_DEBUG_LOGGER DEBUG - 2006-02-26 21:22:53,085 - [LOG_ID 1] LexBIG Logger configured`
	`LB_DEBUG_LOGGER DEBUG - 2006-02-26 21:22:53,085 - [LOG_ID 2] Reading properties from C:\lexbig\resources\config\co`
	`LB_DEBUG_LOGGER DEBUG - 2006-02-26 21:22:53,085 - [LOG_ID 3] Reading registery, index location and db configuratio`
	`LB_DEBUG_LOGGER DEBUG - 2006-02-26 21:22:53,085 - [LOG_ID 4] Reading the Preconfigured SQL Server configurations f`
	`LB_DEBUG_LOGGER DEBUG - 2006-02-26 21:22:53,085 - [LOG_ID 5] Reading the Prebuilt Lucene index configurations from`
	`LB_DEBUG_LOGGER DEBUG - 2006-02-26 21:22:53,085 - [LOG_ID 6] Reading additional variables from the properties file`
	`LB_INFO_LOGGER INFO - 2006-02-26 21:22:53,085 - [LOG_ID 7] Logging debug messages turned off.`

	Open LexBIG_load_log.txt using text editor to review details about vocabulary load utilities.
	
	UltraEdit-32 - [C:\lexbig\admin\logs\LexBIG_full_log.txt]
	`LB_DEBUG_LOGGER DEBUG - 2006-02-26 21:22:53,085 - [LOG_ID 1] LexBIG Logger configured`
	`LB_DEBUG_LOGGER DEBUG - 2006-02-26 21:22:53,085 - [LOG_ID 2] Reading properties from C:\lexbig\resources\config\co`
	`LB_DEBUG_LOGGER DEBUG - 2006-02-26 21:22:53,085 - [LOG_ID 3] Reading registery, index location and db configuratio`
	`LB_DEBUG_LOGGER DEBUG - 2006-02-26 21:22:53,085 - [LOG_ID 4] Reading the Preconfigured SQL Server configurations f`
	`LB_DEBUG_LOGGER DEBUG - 2006-02-26 21:22:53,085 - [LOG_ID 5] Reading the Prebuilt Lucene index configurations from`
	`LB_DEBUG_LOGGER DEBUG - 2006-02-26 21:22:53,085 - [LOG_ID 6] Reading additional variables from the properties file`
	`LB_INFO_LOGGER INFO - 2006-02-26 21:22:53,085 - [LOG_ID 7] Logging debug messages turned off.`

In addition to the log information, system properties are included as part of the system verification test as html or xml format. A sample of the system properties in html is included in Figure 5.


Properties of org.LexGrid.LexBIG.test.AllTests - Microsoft Internet Explorer
Close

Properties of org.LexGrid.LexBIG.test.AllTests

Name	Value
ant.file	C:\lexbig2\test\TestRunner.xml
ant.file.LexBIG TestRunner	C:\lexbig2\test\TestRunner.xml
ant.home	C:\lexbig2\test\extlib
ant.java.version	1.5
ant.library.dir	C:\lexbig2\test\extlib\ant
ant.project.name	LexBIG TestRunner
ant.version	Apache Ant version 1.6.5 compiled on June 2 2005
awt.toolkit	sun.awt.windows.WToolkit
basedir	C:\lexbig2\test
DSTAMP	20060227
file.encoding	Cp1252
file.encoding.pkg	sun.io
file.separator	\
java.awt.graphicsenv	sun.awt.Win32GraphicsEnvironment
java.awt.printerjob	sun.awt.windows.WPrinterJob
java.class.path	.\lbTest.jar;..\runtime\lbRuntime.jar;.\extlib\ant\ant.jar;.\extlib\ant\ant-junit.jar;.\extlib\ant\ant-trax.jar;.\extlib\ant\ant-launcher.jar;.\extlib\junit\junit.jar;C:\lexbig2\test\extlib\ant\ant-junit.jar;C:\ \test\extlib\ant\ant-launcher.jar;C:\lexbig2\test\extlib\ant\ant-trax.jar;C:\le \test\extlib\ant\ant.jar
java.class.version	49.0
java.endorsed.dirs	C:\Program Files\Java\jre1.5.0_06\lib\endorsed
java.ext.dirs	C:\Program Files\Java\jre1.5.0_06\lib\ext
java.home	C:\Program Files\Java\jre1.5.0_06
java.io.tmpdir	C:\DOCUME~1\JDB02~1.MFA\LOCALS~1\Temp\
java.library.path	C:\WINDOWS\system32;.;C:\WINDOWS\system32;C:\WINDOWS;c:\sybase \ODBC;c:\sybase12_5_1\OLEDB;c:\sybase12_5_1\OCS-12_5\bin;c:\sybase \OCS-12_5\dll;C:\sybase12_5_1\ADO.NET\dll;c:\sybase12_5\OLEDB;c:\syb \ODBC;c:\sybase12_5\OCS-12_5\dll;c:\sybase12_5\OCS-12_5\lib3p;c:\syb \OCS-12_5\bin;C:\Perl\bin\;"C:\Program Files\IBM\uima\bin";C:\WINDOWS\system32;C:\WINDOWS;C:\WINDOWS\S \Wbem;C:\PROGRA~1\GNU\WINCVS~1.3 \CVSNT;c:\sybase\bin;c:\sybase\dll;C:\Program Files\Common Files\Adapte

Figure 5 – System Properties from System Verification Test

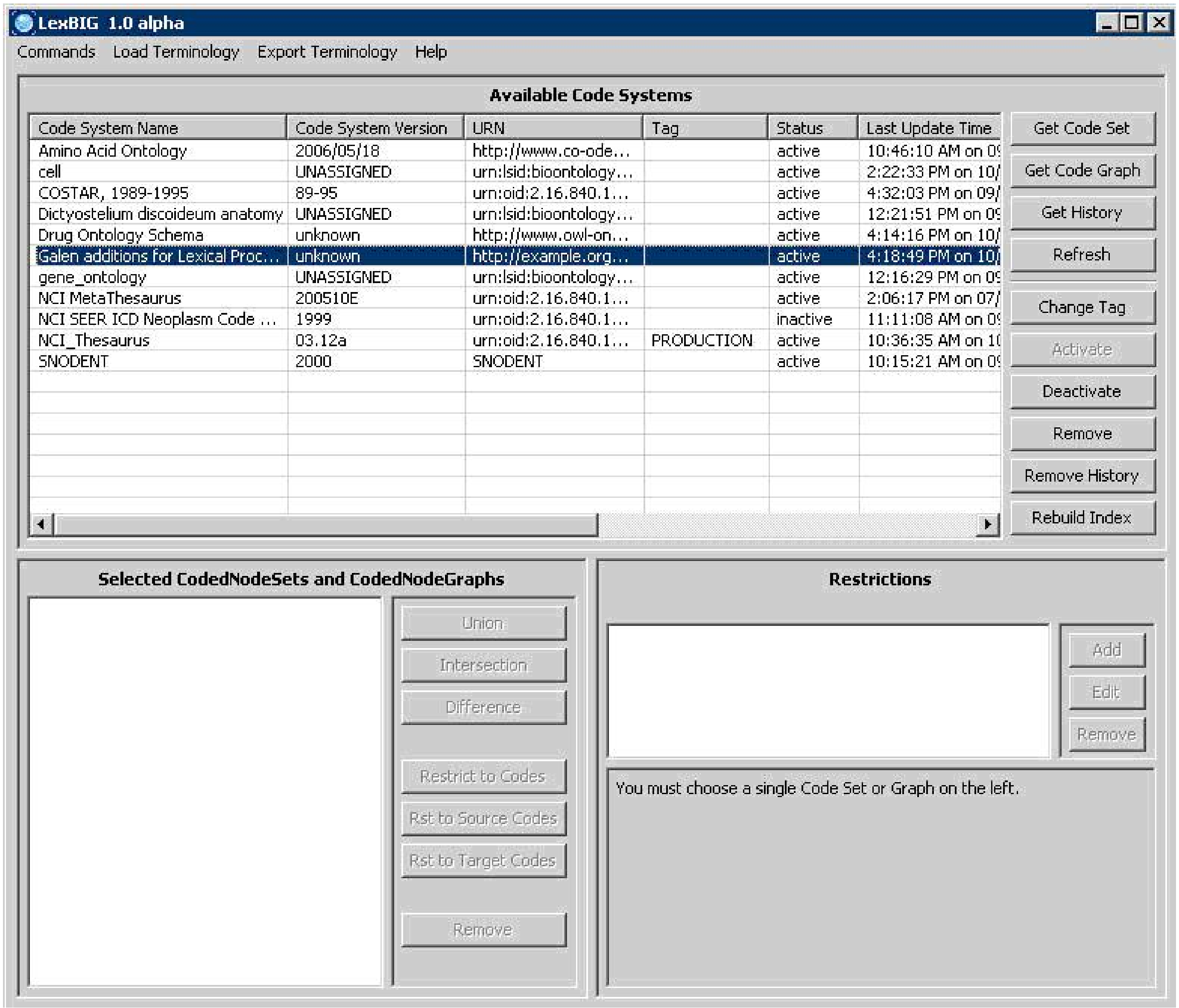
LexEVS GUI Admin Tool

If you choose to install the LexEVS GUI when you installed LexEVS – you will have a `'gui'` folder inside of your LexEVS base installation. If you installed the GUI for all operating systems, you should have the following programs in the `'`
`Linux_64-lbGUI.sh Linux-lbGUI.sh OSX-lbGUI.command Windows-lbGUI.bat Windows-lbGUI -browser.bat`

We provide two Windows shell script versions which allow a choice between the full fledged loading, managing and "end use" or "end use" only type interfaces.

This shell script provides an example by which any shell script can pass an argument option "-d" into the java command launching the LexEVS GUI application, restricting the end user to browsing only and allowing no loading or management of terminologies.

Launch the GUI by executing the appropriate script for your platform. You will be presented with an application that looks like this:



Code System Name	Code System Version	URN	Tag	Status	Last Update Time
Amino Acid Ontology	2006/05/18	http://www.co-ode…		active	10:46:10 AM on 0
cell	UNASSIGNED	urn:lsid:bioontology…		active	2:22:33 PM on 10/
COSTAR, 1989-1995	89-95	urn:oid:2.16.840.1…		active	4:32:03 PM on 09/
Dictyostelium discoideum anatomy	UNASSIGNED	urn:lsid:bioontology…		active	12:21:51 PM on 0
Drug Ontology Schema	unknown	http://www.owl-on…		active	4:14:16 PM on 10/
Galen additions for Lexical Proc…	unknown	http://example.org…		active	4:18:49 PM on 10/
gene_ontology	UNASSIGNED	urn:lsid:bioontology…		active	12:16:29 PM on 0
NCI MetaThesaurus	200510E	urn:oid:2.16.840.1…		active	2:06:17 PM on 07/
NCI SEER ICD Neoplasm Code …	1999	urn:oid:2.16.840.1…		inactive	11:11:08 AM on 0
NCI_Thesaurus	03.12a	urn:oid:2.16.840.1…	PRODUCTION	active	10:36:35 AM on 1
SNODENT	2000	SNODENT		active	10:15:21 AM on 0

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

This application will let you perform most administrative functions that are available in the LexEVS API. To enable the administrative functions, first, go to the 'Commands' menu, and then click on the 'Enable Admin Options' submenu. This will enable all of the commands that can make changes to the LexEVS environment.

This guide will only cover the administrative commands – please refer to the programmers guide for instructions on the rest of the LexEVS GUI.

Each administrative command will be described in turn, starting with the menus.

'Commands' Menu	
SubmenuProperty Name	**Menu Action**
Configure	This menu option will bring up a dialog which will show you all of the options from the current `config.props` file. You can make changes to individual variable here – but these changes will only affect the GUI – they will not be written back out to the `config.props` file. You can also choose which `config.props` file that you want to use.
Enable Admin Options	This option enables or disables all of the GUI features which are considered administrative options.
Clean Up	This command will run the clean up orphaned resources tool. It will give you a listing of any

	resources that are orphaned in the LexEVS environment, and give you the option to remove them.
View Log File	This will show you the all of the logs messages that have occurred during the LexEVS GUI session. The log file viewer also has choices to let you customize the types of messages that are logged.
Exit	Close the application.

'Export Terminology' Menu	
SubmenuProperty Name	**Menu Action**
Export as OBO	This menu option will launch an exporter that exports the selected terminology into an OBO 1.2 format.
Export as LexGrid XML	This menu option will launch an exporter that exports the selected terminology into the LexGrid XML format.

Now that all of the menus have been covered, we will go over the administrative buttons in the LexEVS GUI. These can be found in the lower right area of the top half of the application.

Button	Button Action
Change Tag	Brings up a dialog that allows you to set (or remove) the tag on the selected terminology.
Activate	Activates the selected terminology. Only available if the terminology is currently deactivated.
Deactivate	Deactivates the selected terminology. Only available if the terminology is currently activated.
Remove	Deletes the selected terminology.
Remove History	Removes the NCI History data for the selected terminology. Only applicable to NCI Thesaurus terminologies.
Rebuild Index	Rebuilds the internal indexes for the selected terminology. If no terminology is selected, rebuilds the indexes for all terminologies.

Vocabulary Administration

A set of administrative utilities are provided to manage the LexEVS Service. These utilities are provided for Windows (*.bat) and Linux (*.sh) operating systems. Each of the commands is located in the `{LEXBIG_DIRECTORY}/admin` and `{LEXBIG_DIRECTORY}/test directory.` A full description of the options with example is provided for each of the administration utilities.

Administrative Program	*Description*
ActivateScheme	Activates a coding scheme based on unique URN and version. Options: -u,--urn <urn> URN uniquely identifying the code system. -v,--version <versionId> Version identifier. -f,--force Force activation (no confirmation). Example: ActivateScheme -u "urn:oid:2.16.840.1.113883.3.26.1.1" -v "05.09e"
ClearOrphanedResources	Clean up orphaned resources - databases and indexes. Options: -li,--listIndexes List all unused indexes. -ldb,--listDatabases List all unused databases (with matching prefix). -ri,--removeIndex <name> Remove the (unused) index with the given name. -rdb,--removeDatabase <name> Remove the (unused) database with the given name. -a,--all Remove all unreferenced indexes and databases (with matching prefix). Example: ClearOrphanedResources -li
DeactivateScheme	Deactivates a coding scheme based on unique URN and version. Options: -u,--urn <urn> URN uniquely identifying the code system. -v,--version <versionId> Version identifier. -d,--date <yyyy-MM-dd,HH:mm:ss> Date and time for deactivation to take effect; immediate if not specified. -f,--force Force deactivation (no confirmation).

	Example: DeactivateScheme -u "urn:oid:2.16.840.1.113883.3.26.1.1" -v "05.09e" -d "01/31/2099,12:00:00"
ExportLgXML	Exports content from the repository to a file in the LexGrid canonical XML format. Options: -out,--output <uri> URI or path of the directory to contain the resulting XML file. The file name will be automatically derived from the coding scheme name. -u,--urn <name> URN or local name of the coding scheme to export. -v,--version <id> The assigned tag/label or absolute version identifier of the coding scheme. -nf,--noFail If specified, indicates that processing should not stop for recoverable errors -f,--force If specified, allows the destination file to be overwritten if present. Note: If the coding scheme and version values are unspecified, a list of available coding schemes will be presented for user selection. Example: ExportLgXML -out "file:///path/to/dir" -nf -f Example: ExportLgXML -out "file:///path/to/dir" u "NCI_Thesaurus" -v "PRODUCTION" -nf -f
ExportOBO	Exports content from the repository to a file in the Open Biomedical Ontologies (OBO) format. Options: -out,--output <uri> URI or path of the directory to contain the resulting OBO file. The file name will be automatically derived from the coding scheme name. -u,--urn <name> URN or local name of the coding scheme to export. -v,--version <id> The assigned tag/label or absolute version identifier of the coding scheme. -nf,--noFail If specified, indicates that processing should not stop for recoverable errors -f,--force If specified, allows the destination file to be overwritten if present. Note: If the coding scheme and version values are unspecified, a list of available coding schemes will be presented for user selection. Example: ExportOBO -out "file:///path/to/dir" -nf -f Example: ExportOBO -out "file:///path/to/dir" -u "FBbt" -v "PRODUCTION" -nf -f
ExportOWL	Exports content from the repository to a file in OWL format. Options: -out,--output <uri> URI or path of the directory to contain the resulting OWL file. The file name will be automatically derived from the coding scheme name. -u,--urn <name> URN or local name of the coding scheme to export. -v,--version <id> The assigned tag/label or absolute version identifier of the coding scheme. -nf,--noFail If specified, indicates that processing should not stop for recoverable errors -f,--force If specified, allows the destination file to be overwritten if present. Note: If the URN and version values are unspecified, a list of available coding schemes will be presented for user selection. Example: ExportOWL -out "file:///path/to/dir" -nf -f Example: ExportOWL -out "file:///path/to/dir" -u "sample" -v "1.0" -nf -f
ListExtensions	List registered extensions to the LexEVS runtime environment. Options: -a,--all List all extensions (default, override by specifying other options). -i,--index List index extensions. -m,--match List match algorithm extensions. -s,--sort List sort algorithm extensions. -g,--generic List generic extensions. Example: ListExtensions –a
ListSchemes	List all currently registered vocabularies. Options: -b,--brief List only coding scheme name, version, urn, and tags (default). -f,--full List full detail for each scheme.

	Example: ListSchemes
LoadLgXML	Loads a vocabulary file, provided in LexGrid canonical xml format. Options: -in,--input <uri> URI specifying location of the source file. -v, --validate <level> Perform validation of the candidate resource without loading data. If specified, the '-nf', -a' and '-t' options are ignored. Supported levels of validation include: 0 = Verify document is well-formed 1 = Verify document is valid -nf,--noFail If specified, indicates that processing should not stop for recoverable errors. -a, --activate ActivateScheme on successful load; if unspecified the vocabulary is loaded but not activated. -t, --tag <tagID> An optional tag ID (e.g. 'PRODUCTION' or 'TEST') to assign. Load Example: LoadLgXML -in "file:///path/to/file.xml" -nf –a Validation Example: LoadLgXML -in "file:///path/to/file.xml" -v 0
LoadNCIHistory	Imports NCI History data to the LexEVS repository. Options: -in,--input <uri> URI specifying location of the history file -vf,--versionFile <uri> URI specifying location of the file containing version identifiers for the history to be loaded. -v, --validate <level> Perform validation of the candidate resource without loading data. If specified, the '-nf' and '-r' options are ignored. Supported levels of validation include: 0 = Verify top 10 lines are correct format 1 = Verify correct format for the entire file -nf,--noFail If specified, indicates that processing should not stop for recoverable errors -r, --replace If not specified, the provided history file will be added into the current history database; otherwise the current database will be replaced by the new content. Load Example: LoadNCIHistory –nf -in "file:///path/to/history.file" –vf “file:///path/to/version.file” Validation Example: LoadNCIHistory -in "file:///path/to/history.file" -v 0 Versions File format information: releaseDate|isLatest|releaseAgency|releaseId|releaseOrder|entityDescription Sample record: 28-NOV-05|false|http://nci.nih.gov/|05.10e|26|Editing of NCI Thesaurus 05.10e was completed on October 31, 2005. Version 05.10e was October's fifth build in our development cycle.
LoadNCIMeta	Loads the NCI MetaThesaurus, provided as a collection of RRF files. Options: -in,--input <uri> The directory containing the RRF files; in URI format. -v, --validate <level> Perform validation of the candidate resource without loading data. If specified, the '-nf', -a' and '-t' options are ignored. Supported levels of validation include: 0 = Verify first 1000 lines per required file -nf,--noFail If specified, indicates that processing should not stop for recoverable errors -a, --activate ActivateScheme on successful load; if unspecified the vocabulary is loaded but not activated -t, --tag <tagID> An optional tag ID (e.g. 'PRODUCTION' or 'TEST') to assign. Load Example: LoadNCIMeta -in "file:///path/to/directory" -nf –a Validation Example: LoadNCIMeta -in "file:///path/to/directory" -v 0
LoadNCIThesOWL	Loads an OWL file containing a version of the NCI Thesaurus ... Options: -in,--input <uri> URI specifying location of the source file -v, --validate <level> Perform validation of the candidate resource without loading data. If specified, the '-nf', -a' and '-t' options are ignored. Supported levels of validation include: 0 = Verify document is well-formed 1 = Verify document is valid -nf,--noFail If specified, indicates

	that processing should not stop for recoverable errors. -a, --activate ActivateScheme on successful load; if unspecified the vocabulary is loaded but not activated. -t, --tag <tagID> An optional tag ID (e.g. 'PRODUCTION' or 'TEST') to assign. Load Example: LoadNCIThesOWL -in "file:///path/to/thesaurus.owl" -nf -a Validation Example: LoadNCIThesOWL -in "file:///path/to/thesaurus.owl" -v 0
LoadOBO	Loads a file specified in the Open Biomedical Ontologies (OBO) format. Options: -in,--input <uri> URI or path specifying location of the source file -v, --validate <int> Perform validation of the candidate resource without loading data. If specified, the '-nf', -a' and '-t' options are ignored. Supported levels of validation include: 0 = Verify document is valid -nf,--noFail If specified, indicates that processing should not stop for recoverable errors -a, --activate ActivateScheme on successful load; if unspecified the vocabulary is loaded but not activated -t, --tag <id> An optional tag ID (e.g. 'PRODUCTION' or 'TEST') to assign. Example: LoadOBO -in "file:///path/to/file.obo" -nf -a LoadOBO -in "file:///path/to/file.obo" -v 0
LoadOWL	Loads an OWL file. Note: Load of the NCI Thesaurus should be performed via the LoadNCIThesOWL counterpart, since it will allow more precise handling of NCI semantics. Options: -in,--input <uri> URI or path specifying location of the source file -v, --validate <int> Perform validation of the candidate resource without loading data. If specified, the '-nf', -a' and '-t' options are ignored. Supported levels of validation include: 0 = Verify document is well-formed 1 = Verify document is valid -nf,--noFail If specified, indicates that processing should not stop for recoverable errors -a, --activate ActivateScheme on successful load; if unspecified the vocabulary is loaded but not activated -t, --tag <id> An optional tag ID (e.g. 'PRODUCTION' or 'TEST') to assign. Example: LoadOWL -in "file:///path/to/somefile.owl" -nf -a LoadOWL -in "file:///path/to/somefile.owl" -v 0
LoadUMLSDatabase	Loads UMLS content, provided as a collection of RRF files in a single directory. Files may comprise the entire UMLS distribution or pruned via the MetamorphoSys tool. A complete list of source vocabularies is available online at http://www.nlm.nih.gov/research/umls/metaa1.html. Options: -in,--input <uri> Location of the source database. Typically this is specified in the form of a URL that indicates the database server, port, name, and optional properties. -u,--uid User ID for authenticated access, if required and not specified as part of the input URL. -p,--pwd Password for authenticated access, if required and not specified as part of the input URL. -d,--driver Name of the JDBC driver to use when accessing the database. -s,--sources Comma-delimited list of source vocabularies to load. If absent, all available vocabularies are loaded. -v, --validate <int> Perform validation of the candidate resource without loading data. If specified, the '-nf', -a' and '-t' options are ignored. Supported levels of validation include: 0 = Verify the existence of each required file -nf,--noFail If specified, indicates that processing should not stop for recoverable errors -a, --activate ActivateScheme on successful load; if unspecified the vocabulary is loaded but not activated. -t, --tag <id> An optional tag ID (e.g. 'PRODUCTION' or 'TEST') to assign. Example: LoadUMLSDatabase -in "jdbc:postgresql://localhost:5432/lexgrid" -d "org.postgresql.Driver" -u "myDatabaseUser" -p "myPassword" -s "ICD9CM_2005,ICD9CM_2006" -nf -a LoadUMLSDatabase -in "jdbc:postgresql://localhost:5432/lexgrid" -d "org.postgresql.Driver" -u

	"myDatabaseUser" -p "myPassword" -v 0
LoadUMLSFiles	Loads UMLS content, provided as a collection of RRF files in a single directory. Files may comprise the entire UMLS distribution or pruned via the MetamorphoSys tool. A complete list of source vocabularies is available online at http://www.nlm.nih.gov/research/umls/metaa1.html. Options: -in,--input <uri> URI or path of the directory containing the NLM files -s,--sources Comma-delimited list of source vocabularies to load. If absent, all available vocabularies are loaded. -v, --validate <int> Perform validation of the candidate resource without loading data. If specified, the '-nf', -a' and '-t' options are ignored. Supported levels of validation include: 0 = Verify the existence of each required file -nf,--noFail If specified, indicates that processing should not stop for recoverable errors -a, --activate ActivateScheme on successful load; if unspecified the vocabulary is loaded but not activated. -t, --tag <id> An optional tag ID (e.g. 'PRODUCTION' or 'TEST') to assign. Example: LoadUMLSFiles -in "file:///path/to/directory/" -s "ICD9CM_2005,ICD9CM_2006" -nf -a LoadUMLSFiles -in "file:///path/to/directory/" -v 0 **Note:** UMLS Metathesaurus RRF files are a very large fileset. Many users prefer to subset these files using the Metamorphosys tool included with the UMLS Metathesaurus in order to move a single terminology from a central location of these files. When generating source RRF files from the Metathesaurus, the Metamorphosys tool should be set to output versionless source abbreviations rather than versioned source abbreviations. Failing to do so before loading RRF files to LexEVS will cause an incomplete database to be created leaving the association and concept tables empty
LoadUMLSSemnet	Loads the UMLS Semantic Network, provided as a collection of files in a single directory. The following files are expected to be provided from the National Library of Medicine (NLM) distribution: - LICENSE.txt (text from distribution terms and conditions) - SRFIL.txt (File Description) - SRFIL.txt (Field Description) - SRDEF.txt (Basic information about the Semantic Types and Relations) - SRSTR.txt (Structure of the Network) - SRSTRE1.txt (Fully inherited set of Relations (UIs)) - SRSTRE2.txt (Fully inherited set of Relations (Names)) - SU.txt (Unit Record) These files can be downloaded from the NLM web site at http://semanticnetwork.nlm.nih.gov/Download/index.html. Options: -in,--input <uri> URI or path of the directory containing the NLM files -v, --validate <int> Perform validation of the candidate resource without loading data. If specified, the '-nf', -a' and '-t' options are ignored. Supported levels of validation include: 0 = Verify the existence of each required file -nf,--noFail If specified, indicates that processing should not stop for recoverable errors -a, --activate ActivateScheme on successful load; if unspecified the vocabulary is loaded but not activated. -t, --tag <id> An optional tag ID (e.g. 'PRODUCTION' or 'TEST') to assign. -il --InheritanceLevel <int> If specified, indicates the extent of inherited relationships to import. 0 = none; 1 = all; 2 = all except is_a (default). All direct relationships are imported, regardless of option. Example: LoadUMLSSemnet -in "file:///path/to/directory/" -nf –a –il 1 LoadUMLSSemnet -in "file:///path/to/directory/" -v 0
	Imports from an FMA database to a LexEVS repository. Requires that the pprj file be configured with a database URN, username, password for an FMA MySQL based database. The FMA.pprj file and MySQL dump file are available at http://sig.biostr.washington.edu/projects/fm/ upon registration. Options: -in,--input <uri> URI or path specifying location of the source file -v, --validate <int> Perform validation of the candidate resource without loading data. If specified, the

LoadFMA	'-nf', -a' and '-t' options are ignored. Supported levels of validation include: 0 = Verify document is well-formed 1 = Verify document is valid -nf,--noFail If specified, indicates that processing should not stop for recoverable errors -a, --activate ActivateScheme on successful load; if unspecified the vocabulary is loaded but not activated -t, --tag <id> An optional tag ID (e.g. 'PRODUCTION' or 'TEST') to assign. Example: LoadFMA -in "file:///path/to/FMA.pprj" -nf -a -or- LoadFMA -in "file:///path/to/FMA.pprj" -v 0
LoadHL7RIM	Converts an HL7 RIM MS Access database to a LexGrid database -in,--input <uri> URI or path specifying location of the source file -mf,--manifest <uri> URI or path specifying location of the manifest file -lp,--load preferences <uri> URI or path specifying location of the load preferences file -v, --validate <int> Perform validation of the candidate resource without loading data. If specified, the '-nf', -a' and '-t' options are ignored. Supported levels of validation include: 0 = Verify document is valid -nf,--noFail If specified, indicates that processing should not stop for recoverable errors -a, --activate ActivateScheme on successful load; if unspecified the vocabulary is loaded but not activated -t, --tag <id> An optional tag ID (e.g. 'PRODUCTION' or 'TEST') to assign. Example: LoadHL7RIM -in "file:///path/to/file.mdb" -nf -a -or- LoadHL7RIM -in "file:///path/to/file.mdb" -v 0
LoadMetaData	Loads optional XML-based metadata to be associated with an existing coding scheme. -u,--urn <name> URN uniquely identifying the code system. -v,--version <id> Version identifier. -in,--input <uri> URI or path specifying location of the XML file. -v, --validate <int> Perform validation of the input file without loading data. If specified, the '-nf', '-f', and '-o' options are ignored. Supported levels of validation include: 0 = Verify document is valid -o, --overwrite If specified, existing metadata for the code system will be erased. Otherwise, new metadata will be appended to existing metadata (if present). -f,--force Force overwrite (no confirmation). -nf,--noFail If specified, indicates that processing should not stop for recoverable errors Note: If the URN and version values are unspecified, a list of available coding schemes will be presented for user selection. Example: LoadMetadata -in "file:///path/to/file.xml" -nf -o -or- LoadMetadata -in "file:///path/to/file.xml"
RebuildIndex	Rebuilds indexes associated with the specified coding scheme. Options: -u,--urn <urn> URN uniquely identifying the code system. -v,--version <versionId> Version identifier. -i,--index <name> Name of the index extension to rebuild (if absent, rebuilds all built-in indices and named extensions). -f,--force Force clear (no confirmation). Example: RebuildIndex -u "urn:oid:2.16.840.1.113883.3.26.1.1" -v "05.09e" -i "myindex"
RemoveIndex	Clears an optional named index associated with the specified coding scheme. Note: built-in indices required by the LexEVS runtime cannot be removed. Options -u,--urn <urn> URN uniquely identifying the code system. -v,--version <versionId> Version identifier. -i,--index <name> Name of the index extension to clear. -f,--force Force clear (no confirmation). Example: RemoveIndex -u "urn:oid:2.16.840.1.113883.3.26.1.1" -v "05.09e" -i "myindex"
	Removes a coding scheme based on unique URN and version.

RemoveScheme	Options: -u,--urn <urn> URN uniquely identifying the code system. -v,--version <versionId> Version identifier. -f,--force Force deactivation and removal without confirmation. Example: RemoveScheme -u "urn:oid:2.16.840.1.113883.3.26.1.1" -v "05.09e"
TagScheme	Associates a tag ID (e.g. 'PRODUCTION' or 'TEST') with a coding scheme URN and version. Options: -u,--urn <urn> URN uniquely identifying the code system. -v,--version <versionId> Version identifier. -t,--tag The tag ID (e.g. 'PRODUCTION' or 'TEST') to assign. Example: TagScheme -u "urn:oid:2.16.840.1.113883.3.26.1.1" -v 05.09e" -t "TEST"
TestRunner* *Located in {LEXBIG_DIRECTORY}/test **Note:** the LexEVS runtime and database environments must still be configured prior to invoking the test suite.	Executes a suite of tests for the LexEVS installation. Options: -b,--brief Runs the LexEVS test suite and produce a text report with overall statistics and details for failed tests only. -f,--full Runs the LexEVS test suite and produce an itemized list of all tests with indication of success/failure. -h,--html Runs the LexEVS test suite and produce a report suitable for view in a standard web browser. -x,--xml Runs the LexEVS test suite and produce a report with extensive information for each test case in xml format. Example: TestRunner –f -h
TransferScheme	Tool to help gather information necessary to transfer data from one SQL server to another. Options: -u,--urn The Coding Scheme URN or local name to transfer. -v,--version The version of the coding scheme to transfer. Example: TransferScheme -u "urn:oid:2.16.840.1.113883.3.26.1.1" -v 05.09e"

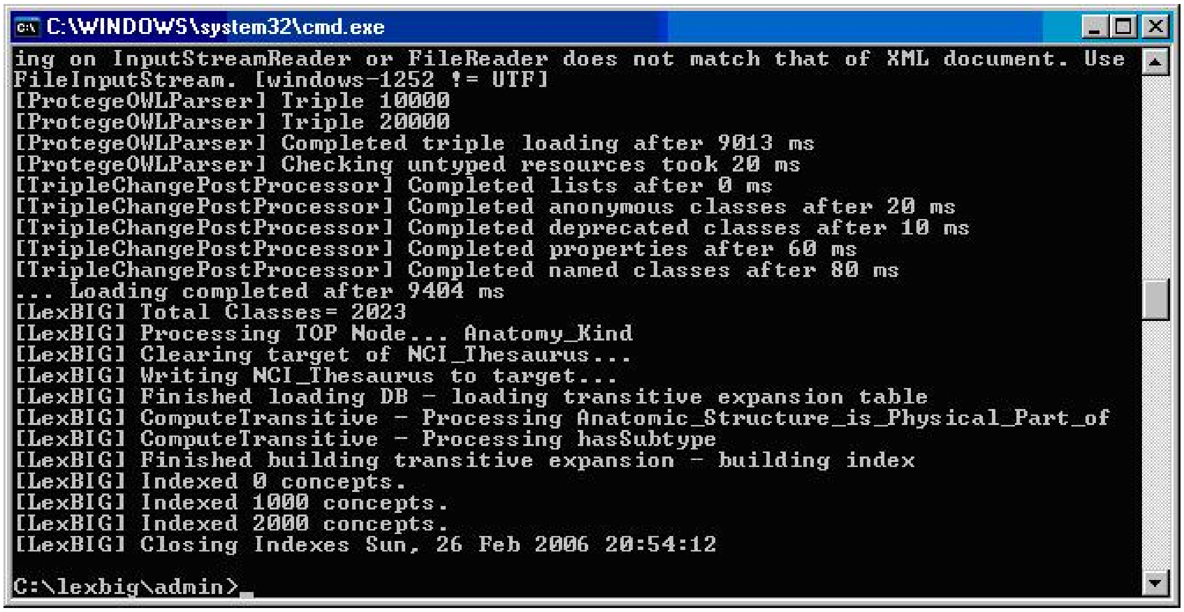
Loading Vocabularies

Samples

This LexEVS installation provides the UMLS Semantic Net and a sampling of the NCI Thesaurus content (sample.owl) that can be loaded into the database.

Step	Action
1	In a **Command Prompt** window, enter ***`cd {LEXBIG_DIRECTORY}/examples`*** to go to the example programs.
2	To load the example vocabularies, run the appropriate LoadSampleData script (`LoadSampleData.bat` for Windows; `LoadSampleData.sh` for Linux).

NOTE:	Vocabularies should not be loaded until configuration of the LexEVS runtime and database server are complete.



```
ing on InputStreamReader or FileReader does not match that of XML document. Use
FileInputStream. [windows-1252 != UTF]
[ProtegeOWLParser] Triple 10000
[ProtegeOWLParser] Triple 20000
[ProtegeOWLParser] Completed triple loading after 9013 ms
[ProtegeOWLParser] Checking untyped resources took 20 ms
[TripleChangePostProcessor] Completed lists after 0 ms
[TripleChangePostProcessor] Completed anonymous classes after 20 ms
[TripleChangePostProcessor] Completed deprecated classes after 10 ms
[TripleChangePostProcessor] Completed properties after 60 ms
[TripleChangePostProcessor] Completed named classes after 80 ms
... Loading completed after 9404 ms
[LexBIG] Total Classes= 2023
[LexBIG] Processing TOP Node... Anatomy_Kind
[LexBIG] Clearing target of NCI_Thesaurus...
[LexBIG] Writing NCI_Thesaurus to target...
[LexBIG] Finished loading DB - loading transitive expansion table
[LexBIG] ComputeTransitive - Processing Anatomic_Structure_is_Physical_Part_of
[LexBIG] ComputeTransitive - Processing hasSubtype
[LexBIG] Finished building transitive expansion - building index
[LexBIG] Indexed 0 concepts.
[LexBIG] Indexed 1000 concepts.
[LexBIG] Indexed 2000 concepts.
[LexBIG] Closing Indexes Sun, 26 Feb 2006 20:54:12

C:\lexbig\admin>_
```

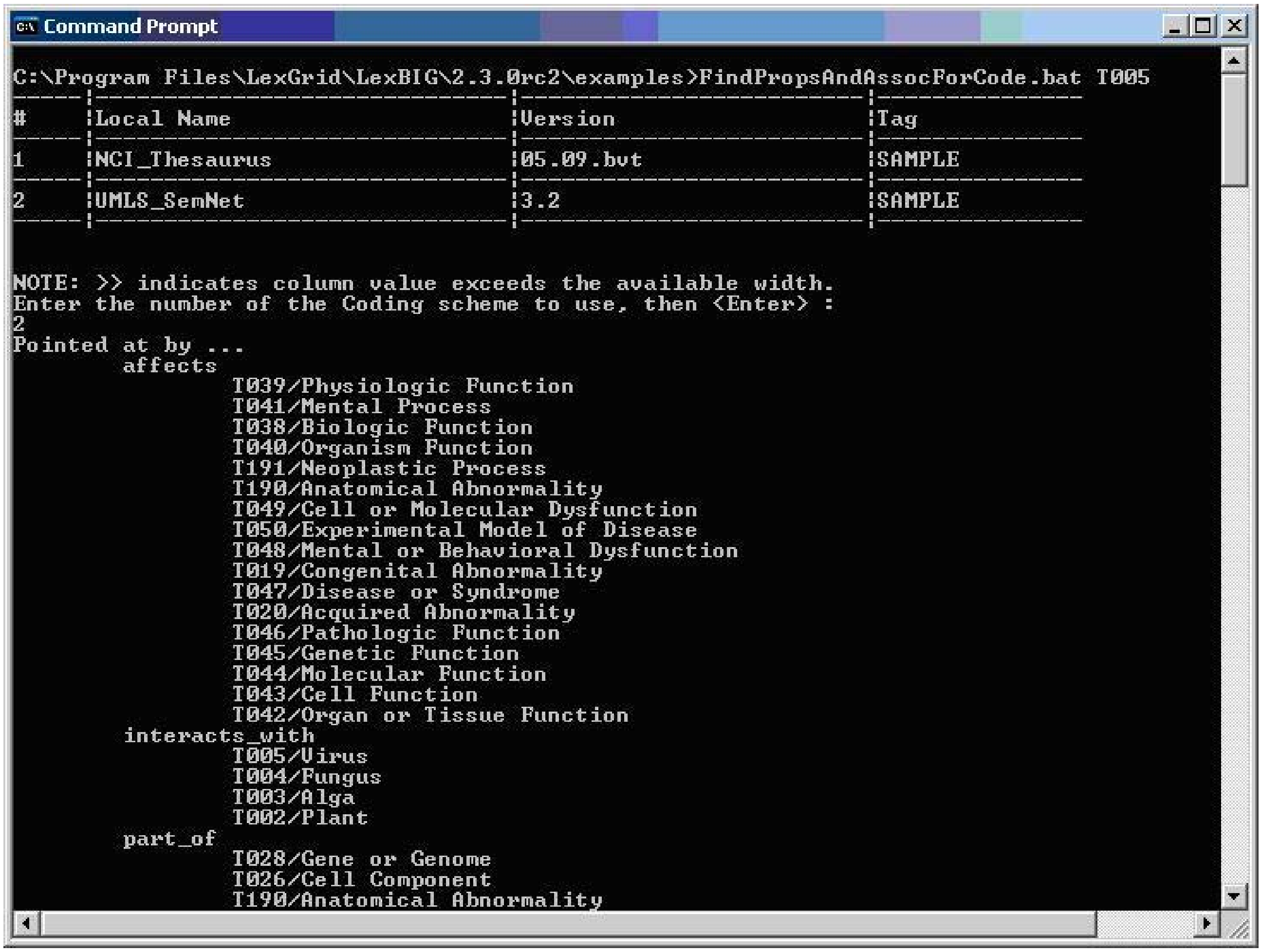
Figure 3 - Displays the successful load of the sample vocabulary file.

Running the Sample Query Programs

A set of sample programs are provided in the {LEXBIG_DIRECTORY}/examples directory. To run the sample query programs successfully a vocabulary must have been loaded.

Step	Action
1	Enter ***cd {LEXBIG_DIRECTORY}/examples***
2	Execute one of sample programs. .bat for windows or .sh for Linux. 1. ***FindConceptNameForCode.bat*** 2. ***FindPropsandAssocForCode.bat*** `<Code>` 3. ***FindRelatedCodes*** `<Code>` 4. ***FindTreeforCodeAndAssoc*** `<Code>`

Figure 4 - Sample program output for finding properties and associations for a given code.



```
C:\Program Files\LexGrid\LexBIG\2.3.0rc2\examples>FindPropsAndAssocForCode.bat T005
-----¦------------------------------¦--------------------------¦---------------
#    ¦Local Name                    ¦Version                   ¦Tag
-----¦------------------------------¦--------------------------¦---------------
1    ¦NCI_Thesaurus                 ¦05.09.bvt                 ¦SAMPLE
-----¦------------------------------¦--------------------------¦---------------
2    ¦UMLS_SemNet                   ¦3.2                       ¦SAMPLE
-----¦------------------------------¦--------------------------¦---------------


NOTE: >> indicates column value exceeds the available width.
Enter the number of the Coding scheme to use, then <Enter> :
2
Pointed at by ...
        affects
                T039/Physiologic Function
                T041/Mental Process
                T038/Biologic Function
                T040/Organism Function
                T191/Neoplastic Process
                T190/Anatomical Abnormality
                T049/Cell or Molecular Dysfunction
                T050/Experimental Model of Disease
                T048/Mental or Behavioral Dysfunction
                T019/Congenital Abnormality
                T047/Disease or Syndrome
                T020/Acquired Abnormality
                T046/Pathologic Function
                T045/Genetic Function
                T044/Molecular Function
                T043/Cell Function
                T042/Organ or Tissue Function
        interacts_with
                T005/Virus
                T004/Fungus
                T003/Alga
                T002/Plant
        part_of
                T028/Gene or Genome
                T026/Cell Component
                T190/Anatomical Abnormality
```

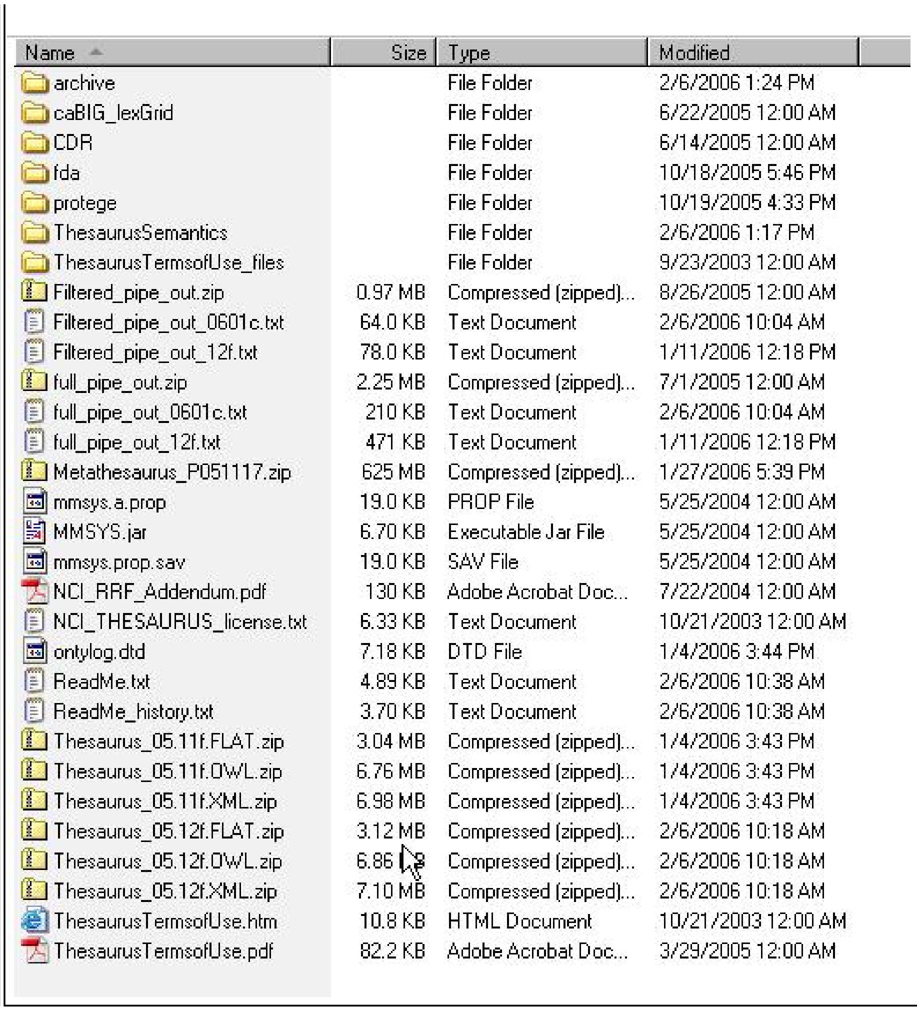
Figure 4 - Output of example programs using sample vocabulary

NCI Vocabularies

Installing NCI Thesaurus Vocabulary

This section describes the steps to download and install a full version of the NCI Thesaurus for the LexEVS Service.

Step	Action
1	Using a web or ftp client go to URL: ftp://ftp1.nci.nih.gov/pub/cacore/EVS/



Name	Size	Type	Modified
archive		File Folder	2/6/2006 1:24 PM
caBIG_lexGrid		File Folder	6/22/2005 12:00 AM
CDR		File Folder	6/14/2005 12:00 AM
fda		File Folder	10/18/2005 5:46 PM
protege		File Folder	10/19/2005 4:33 PM
ThesaurusSemantics		File Folder	2/6/2006 1:17 PM
ThesaurusTermsofUse_files		File Folder	9/23/2003 12:00 AM
Filtered_pipe_out.zip	0.97 MB	Compressed (zipped)...	8/26/2005 12:00 AM
Filtered_pipe_out_0601c.txt	64.0 KB	Text Document	2/6/2006 10:04 AM
Filtered_pipe_out_12f.txt	78.0 KB	Text Document	1/11/2006 12:18 PM
full_pipe_out.zip	2.25 MB	Compressed (zipped)...	7/1/2005 12:00 AM
full_pipe_out_0601c.txt	210 KB	Text Document	2/6/2006 10:04 AM
full_pipe_out_12f.txt	471 KB	Text Document	1/11/2006 12:18 PM
Metathesaurus_P051117.zip	625 MB	Compressed (zipped)...	1/27/2006 5:39 PM
mmsys.a.prop	19.0 KB	PROP File	5/25/2004 12:00 AM
MMSYS.jar	6.70 KB	Executable Jar File	5/25/2004 12:00 AM
mmsys.prop.sav	19.0 KB	SAV File	5/25/2004 12:00 AM
NCI_RRF_Addendum.pdf	130 KB	Adobe Acrobat Doc...	7/22/2004 12:00 AM
NCI_THESAURUS_license.txt	6.33 KB	Text Document	10/21/2003 12:00 AM
ontylog.dtd	7.18 KB	DTD File	1/4/2006 3:44 PM
ReadMe.txt	4.89 KB	Text Document	2/6/2006 10:38 AM
ReadMe_history.txt	3.70 KB	Text Document	2/6/2006 10:38 AM
Thesaurus_05.11f.FLAT.zip	3.04 MB	Compressed (zipped)...	1/4/2006 3:43 PM
Thesaurus_05.11f.OWL.zip	6.76 MB	Compressed (zipped)...	1/4/2006 3:43 PM
Thesaurus_05.11f.XML.zip	6.98 MB	Compressed (zipped)...	1/4/2006 3:43 PM
Thesaurus_05.12f.FLAT.zip	3.12 MB	Compressed (zipped)...	2/6/2006 10:18 AM
Thesaurus_05.12f.OWL.zip	6.86 MB	Compressed (zipped)...	2/6/2006 10:18 AM
Thesaurus_05.12f.XML.zip	7.10 MB	Compressed (zipped)...	2/6/2006 10:18 AM
ThesaurusTermsofUse.htm	10.8 KB	HTML Document	10/21/2003 12:00 AM
ThesaurusTermsofUse.pdf	82.2 KB	Adobe Acrobat Doc...	3/29/2005 12:00 AM

2 Select the version of NCI Thesaurus OWL you wish to download. Save the file to a directory on your machine.

3 Extract the OWL file from the zip download and save in a directory on your machine. This directory will be referred to as NCI_THESAURUS_DIRECTORY

4 Using the LexEVS utilities load the NCI Thesaurus

```
cd {LexBIG_DIRECTORY}/admin
```

For Windows installation use the following command

```
LoadNCIThesOWL.bat –nf –in "file:///{NCI_THESAURUS_DIRECTORY}/Thesaurus_05.12f.owl"
```

For Linux installation use the following command

```
LoadNCIThesOWL.sh –nf –in "file:///{NCI_THESAURUS_DIRECTORY}/Thesaurus_05.12f.owl"
```

NOTE:

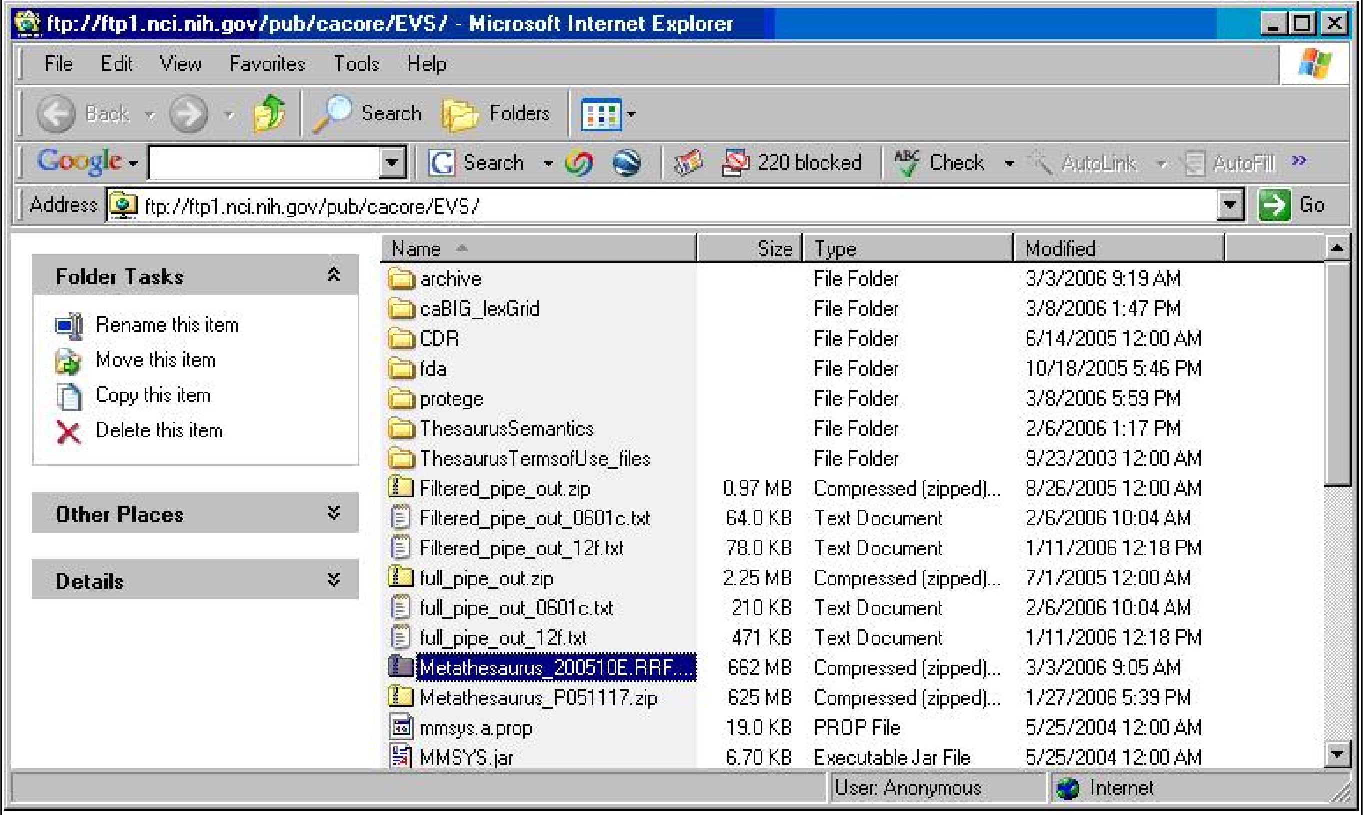
This step will require about three hours on a Pentium 3.0 Ghz machine. The total time to load NCI Thesaurus will vary depending on machine, memory, and disk speed.

Table 7 – Example output from load of NCI Thesaurus 05.12f

```
…
[LexBIG] Processing TOP Node... Retired_Kind
[LexBIG] Clearing target of NCI_Thesaurus...
[LexBIG] Writing NCI_Thesaurus to target...
[LexBIG] Finished loading DB - loading transitive expansion table
[LexBIG] ComputeTransitive - Processing Anatomic_Structure_Has_Location
[LexBIG] ComputeTransitive - Processing Anatomic_Structure_is_Physical_Part_of
[LexBIG] ComputeTransitive - Processing Biological_Process_Has_Initiator_Process
[LexBIG] ComputeTransitive - Processing Biological_Process_Has_Result_Biological_Process
[LexBIG] ComputeTransitive - Processing Biological_Process_Is_Part_of_Process
[LexBIG] ComputeTransitive - Processing Conceptual_Part_Of
[LexBIG] ComputeTransitive - Processing Disease_Excludes_Finding
[LexBIG] ComputeTransitive - Processing Disease_Has_Associated_Disease
[LexBIG] ComputeTransitive - Processing Disease_Has_Finding
[LexBIG] ComputeTransitive - Processing Disease_May_Have_Associated_Disease
[LexBIG] ComputeTransitive - Processing Disease_May_Have_Finding
[LexBIG] ComputeTransitive - Processing Gene_Product_Has_Biochemical_Function
[LexBIG] ComputeTransitive - Processing Gene_Product_Has_Chemical_Classification
[LexBIG] ComputeTransitive - Processing Gene_Product_is_Physical_Part_of
[LexBIG] ComputeTransitive - Processing hasSubtype
[LexBIG] Finished building transitive expansion - building index
[LexBIG] Getting a results from sql (a page if using mysql)
[LexBIG] Indexed 0 concepts.
[LexBIG] Indexed 5000 concepts.
[LexBIG] Indexed 10000 concepts.
[LexBIG] Indexed 15000 concepts.
[LexBIG] Indexed 20000 concepts.
[LexBIG] Indexed 25000 concepts.
[LexBIG] Indexed 30000 concepts.
[LexBIG] Indexed 35000 concepts.
[LexBIG] Indexed 40000 concepts.
[LexBIG] Indexed 45000 concepts.
[LexBIG] Indexed 46000 concepts.
[LexBIG] Getting a results from sql (a page if using mysql)
[LexBIG] Closing Indexes Mon, 27 Feb 2006 01:44:22
[LexBIG] Finished indexing
```

Installing NCI Metathesaurus Vocabulary

This section describes the steps to download and install a full version of the NCI Metathesaurus for the LexEVS Service.

Step	Action
1	Using a web or ftp client go to URL: ftp://ftp1.nci.nih.gov/pub/cacore/EVS/ 

Address: ftp://ftp1.nci.nih.gov/pub/cacore/EVS/

Name	Size	Type	Modified
archive		File Folder	3/3/2006 9:19 AM
caBIG_lexGrid		File Folder	3/8/2006 1:47 PM
CDR		File Folder	6/14/2005 12:00 AM
fda		File Folder	10/18/2005 5:46 PM
protege		File Folder	3/8/2006 5:59 PM
ThesaurusSemantics		File Folder	2/6/2006 1:17 PM
ThesaurusTermsofUse_files		File Folder	9/23/2003 12:00 AM
Filtered_pipe_out.zip	0.97 MB	Compressed (zipped)...	8/26/2005 12:00 AM
Filtered_pipe_out_0601c.txt	64.0 KB	Text Document	2/6/2006 10:04 AM
Filtered_pipe_out_12f.txt	78.0 KB	Text Document	1/11/2006 12:18 PM
full_pipe_out.zip	2.25 MB	Compressed (zipped)...	7/1/2005 12:00 AM
full_pipe_out_0601c.txt	210 KB	Text Document	2/6/2006 10:04 AM
full_pipe_out_12f.txt	471 KB	Text Document	1/11/2006 12:18 PM
Metathesaurus_200510E.RRF...	662 MB	Compressed (zipped)...	3/3/2006 9:05 AM
Metathesaurus_P051117.zip	625 MB	Compressed (zipped)...	1/27/2006 5:39 PM
mmsys.a.prop	19.0 KB	PROP File	5/25/2004 12:00 AM
MMSYS.jar	6.70 KB	Executable Jar File	5/25/2004 12:00 AM

2	Select the version of NCI Metathesaurus RRF you wish to download. Save the file to a directory on your machine.
3	Extract the RRF files from the zip download and save in a directory on your machine. This directory will be referred to as NCI_METATHESAURUS_DIRECTORY. Note: RELASE_INFO.RRF is required to be present for the load utility to work.
4	Using the LexEVS utilities load the NCI Thesaurus ***cd {LexBIG_DIRECTORY}/admin*** For Windows installation use the following command ***LoadNCIMeta.bat –nf –in "file:///{NCI_METATHESAURUS_DIRECTORY}/"*** For Linux installation use the following command ***LoadNCIMeta.sh –nf –in "file:///{NCI_THESAURUS_DIRECTORY}/"***

NOTE:	NCI Metathesaurus contains many individual vocabularies and requires several hours to load and index. This step requires about 15 hours on a Pentium 3.0 Ghz machine with 7200rpm disk. The total time to load NCI MetaThesaurus will vary depending on machine, memory, and disk speed.

Installing NCI History Information

This section describes the steps to download and install a history file for NCI Thesaurus.

Step	Action
1	Using a web or ftp client go to URL: ftp://ftp1.nci.nih.gov/pub/cacore/EVS/
2	Select the version of NCI History you wish to download. Save the file to a directory on your machine. Select the VersionFile download to the same directory as the history file.
3	Extract the History files from the zip download and save in a directory on your machine. This directory will be referred to as NCI_HISTORY_DIRECTORY
4	Using the LexEVS utilities load the NCI Thesaurus ***cd {LexBIG_DIRECTORY}/admin*** For Windows installation use the following command ***LoadNCIHistory.bat –nf –in "file:///{NCI_HISTORY_DIRECTORY}" –vf “file:///NCI_HISTORY_DIRECTORY}/VersionFile”*** For Linux installation use the following command ***LoadNCIHistory.sh –nf –in "file:///{NCI_HISTORY_DIRECTORY}" –vf “file:///NCI_HISTORY_DIRECTORY}/VersionFile”***

NOTE:	If a 'releaseId' occurs twice in the file, the last occurrence will be stored. If LexEVS already knows about a releaseId (from a previous history load), the information is updated to match what is provided in the file. This file has to be provided to the load API on every load because you will need to maintain it in the future as each new release is made. We have created this file that should be valid as of today from the information that we found in the archive folder on your ftp server. You can find this file in the 'resources' directory of the LexEVS install.

Deactivating and Removing Vocabulary

This section describes the steps to deactivate a coding scheme and remove coding scheme from LexEVS Service.

Step	Action
1	Change directory to LexEVS administration directory Enter ***`cd {LEXBIG_DIRECTORY}/admin`***
2	Use the DeactiveScheme utility to prevent access to coding scheme. Once a coding scheme is deactivated, client programs will not be able to access the content for the specific coding scheme and version. Example: DeactivateScheme -u "urn:oid:2.16.840.1.113883.3.26.1.1" -v "05.12f”
3	Use RemoveScheme utility to remove coding scheme from LexEVS service and database. Example: RemoveScheme -u "urn:oid:2.16.840.1.113883.3.26.1.1" -v "05.12f"

Tagging a vocabulary

This section describes the steps to tag a coding scheme to be used via LexEVS API.

Step	Action
1	Change directory to LexEVS administration directory Enter ***`cd {LEXBIG_DIRECTORY}/admin`***
2	Use the TagScheme to tag a coding system and version with a local tag name (e.g. PRODUCTION). This tag name can be used via LexEVS API for query restriction. Example: TagScheme -u "urn:oid:2.16.840.1.113883.3.26.1.1" -v "05.12f” -t “PRODUCTION“

Retrieved from "https://cabig-kc.nci.nih.gov/Vocab/KC/index.php/LexEVS_5.0_Administration_Guide"

- This page was last modified on 11 May 2009, at 20:40.