LexEVS 6.x Loader Use Guide

Contents of this Page

LexEVS 6.x Loader Links

- Loader Use Guide
- Loader Guide for developers
  - Included Loaders
  - Model Element Mapping
  - Loader Framework
- LexEVS 6.0 Main Page
- LexEVS Current Release

When first installed, LexEVS (an open-source, enterprise-wide terminology server) comes with no terminologies loaded into it. This documentation will cover the means for loading most content types that can be loaded. LexEVS was built to accommodate a wide variety of input and meld it into a common form - unifying many common source formats.

If you are interested in the underlying details about loading like the LexEVS information model, the loader code itself, and the framework used to create new loaders then you'll want to read the LexEVS 6.x Loader Guide.

The fact that there are many source formats necessitates a variety of LexEVS loaders, each used on a specific incoming source format. When we speak of these generically, inputs to be loaded are typically called **terminologies** or **coding schemes**. Each terminology will have a specific **source format**. For example, if you load the terminology called the "NCI Thesaurus" you will download a file in one of several source formats namely a text file (TXT), a Web Ontology Language file (OWL), or a LexGrid XML file (XML).

LexEVS provides both a LexEVS administrative GUI and LexEVS loader commands to load terminologies. While the LexEVS administrative GUI is very functional, a system administrator may prefer the command line interface because command scripts can be adjusted to increase memory and tune other java virtual machine settings to insure that loads of larger terminologies have adequate resources. For example, a user may select a loading script, open it in an editor, increase the java heap size and PermGen memory, depending on the machine's resources, and save the script before running with the appropriate options written into the command line. Still, the GUI can be convenient for loading smaller terminologies and, in many cases, works fine for loading moderately large terminologies like the NCI Thesaurus. Loading terminologies requires some knowledge of the source format of the terminology.

Generic loading

Most terminology loads can be easily accomplished by pointing either the LexEVS commands or the LexEVS administrative GUI at the terminology source file and running the loader. Before you try this, you should look to see if the terminology you want to load is a special case (at the bottom of this page). Generic loading instructions can be found for the LexEVS administrative GUI or the LexEVS loader commands. For many source formats you can use a variation of the following LexEVS command:

Linux

```
./LoadOWL.sh -in "file:///ontologies/owl/amino-acid.owl"
```

Windows

```
LoadOWL.bat -in "file:///ontologies/owl/amino-acid.owl"
```

This LexEVS loader command loads input in OWL format. Substituting the matching LexEVS loader command for the source format being used and pointing the loader to a local file will load most terminologies. For example, the OBO source format would be loaded by the LoadOBO command. In the LexEVS administrative GUI, all of the various source format loaders are found under the "Load Terminology" menu. The administrative options must be enabled first in the Command menu.

Best Practices

As you work with terminologies in LexEVS there are some things that you will find are the best way to approach things to make life easy as far as loading goes.

Indexing terminologies

Loading any terminologies can be very time consuming and resource intensive and this can be helped by the following recommendations for database optimization. This is more necessary the larger the terminology gets. The LexEVS configuration file, <LEXEVS_HOME>/resources/config/lbconfig.props, should be changed depending on how the primary key for the database should be generated. The default setting for the value of the database primary key is the following:

The default for the `DB_PRIMARY_KEY_STRATEGY` property changed in LexEVS 6.0.3 to `SEQUENTIAL_INTEGER`. If you are at 6.0.3 or later your performance will not be impacted by leaving the default setting. You will need to change the property only if you want the advantage of globally unique IDs.

```
# DB_PRIMARY_KEY_STRATEGY indicates which strategy will be used
# for the primary key of the database tables.
# WARNING - This cannot be changed after the initial
# schema installation.
#
# Allowable values include:
#
#        "GUID"
#                - Primary Keys are implemented as random GUIDs.
#        "SEQUENTIAL_INTEGER"
#                - Primary Keys will be sequentially incremented
#                - as Ingeter values.
DB_PRIMARY_KEY_STRATEGY=GUID
```

Because this default is very taxing on the index processing at the end of the load, we recommend changing it to SEQUENTIAL_INTEGER for any terminology unless you have a priority need for Global Unique Identifiers. Note that this setting is final once any terminology is loaded. You can not change this after it is in effect. Even launching any LexEVS administrative command or opening the LexEVS administrative GUI will make this permanent. The only way to start over and change the setting is to change the lbconfig.props file, drop the database created for LexEVS, and recreate the database. If you are going to make the change this setting then do so before you do anything with LexEVS.

Database Connection setting for loading large terminologies

When large terminologies like NCI Metathesaurus is being loaded, the database connection might get suspended while waiting for the lucene indexing to complete. To mitigate this problem set the autoReconnect to true in the database URL in the lbconfig.props file. If it is MySQL, the URL would look like:

DB_URL=jdbc:mysql://bmidev4:3307/testLoad2010?autoReconnect=true

and you will have to restart MySQL database using following parameters:
--tmpdir /data/mysqltmp
--wait_timeout=100000

The tmpdir could be pointing to any directory but make sure it has enough space (in case NCI Metathesaurus, around 20Gb will be ideal). The --wait_timeout=100000 which is the number of seconds for 27 hours, will hold the connection for 27 hours.

Monitoring a load

While terminologies are being loaded, you can monitor the progress using the LexEVS logs (both 'load' and 'full' log) and if using MySQL, use INNODB tools to monitor Inserts per second. (SHOW INNODB STATUS)

Setting a terminology as active

When you first load any terminology is is not active by default. One thing you must do is to activate it after it is loaded if you want any queries to work against it. The LexEVS Administrative GUI has a button to activate or deactivate any given terminology. All the LexEVS loader commands also have a flag that can set a terminology to be active upon successful load. The reason for having these states is that you can take terminologies offline without having to unload them.

Setting a terminology as the production terminology

It is best if you always tell LexEVS if a terminology is the default, even if you only have one copy of it loaded. Some queries, like queries to terminology metadata, do not work without setting a terminology as the production copy. You do this by tagging a terminology. The LexEVS Administrative GUI has a button to change the tag of any loaded terminology. The LexEVS loader commands have a flag that can be used to set the tag. The tag is a simple string. You can assign any tag you want, but the string recognized by LexEVS is "PRODUCTION" (all caps, no quotes). You should get in the habit of marking loaded terminologies as PRODUCTION. The opposite of that might be "TEST" or just left blank. The reason for having the tagging function is to allow for multiple versions of the same terminology to be loaded with no ill effect because one of them can be designated as the default with this approach.

Restarting distributed services after loading

After loading terminologies in a LexEVS Distributed environment you will not see any results of doing so until you restart the web container. This is a limitation of the LexEVS Distributed service. Get used to restarting the application server after loading any number of terminologies. You do not have to restart after each one.

Special Case Loading

Some terminologies are special cases and need special handling. Each of these has its own documentation:

Special Case	How to handle
Installing NCI Vocabularies in LexEVS 6.x	The NCI Thesaurus differs from other OWL formatted resources and as a result you should follow this documentation. The NCI Metathesaurus is the largest terminology to be loaded and as such it also requires special handling.
Installing OWL Formatted terminologies	OWL terminologies do not normally require special handling, but LexEVS offers some advanced loading options users may take advantage of.
Installing Vocabularies from the UMLS Metathesauruss (RRF)	Terminologies in RRF format typically come from the National Library of Medicine's (NLM) Unified Medical Language System (UMLS). Many terminologies are a subset of the UMLS such as LOINC, SNOMED, MedDRA, HUGO, GO, and ICD to name a few. The terminology you're interested in is a subset of the UMLS if: • The terminology is documented by the NLM as a source for the UMLS or • The "Download" column on the Vocabulary Knowledge Center's Index of Terminologies includes RRF or ◦ The FORMAT type listed in BioPortal is RRF.
Loading Asserted Value Set Definitions and Indexes	Source Asserted Value Set Loads require that value set definitions based on source asserted value sets be loaded. Supporting Indexes for searches on these value sets also need a separate load from the source terminology load (At the moment this is focused on the NCI Thesaurus.)
Loading the Graph Database into ArangoDb for a Given Terminology	LexEVS supports use of a graphing database, ArangoDb for high speed graph resolution of its relationships. These can be loaded from any terminology in the current LexEVS terminology service. The graphs are loaded from the table of associations persisted into the LexGrid data model in MySQL and can be accessed from either a REST graph service or a client service the NodeGraphResolutionService. It is meant to supplement, but not replace, the LexEVS CodedNodeGraph API.
None of the above match and you cannot find a suitable source format.	Many terminology providers produce more than one source format that can be downloaded. Source formats such as text only, CSV, tab delimited, and spreadsheets are not acceptable source formats for LexEVS. If you can not find an acceptable source format for a terminology to load into LexEVS then one option is to download versions that have been placed on BioPortal. These may not be the latest versions available but are easy to download. If you end up with a with a terminology that is not one of the special cases above then you should return to the generic loading instructions.

Common Errors When Loading

Error	Remedy or Indication
Out of Memory Error Heap Error Perm Gen Error	Generally memory related errors that indicate the heap space and/or perm gen space need to be increased when starting the Java VM.
Data Truncation Error	Source formats change over the years and sometimes this results in this database related error. It indicates that a column size is too small for whatever element has been pulled from the source file. This will require an update to LexEVS to fix and a temporary workaround can be to edit the source so that the element is made short enough to fit the database column.

Data Base Connection Error	Check to see if the DBMS is up and running, that your database exists, that the connection parameters are correct, and that proper privileges exist for the connecting user. This error may manifest itself as DAO related errors being generated by the spring framework at some levels of execution in the logs, including failing to create a dao list.
Too Many Files Open	Linux system error requires setting of system properties. This happens when the Lucene index is large and is being reindexed on loading a new terminology or is being optimized or cleaned up.
Loader Hangs with no Errors	This can happen when there is processor capacity is maxed out, or there is network latency of one variety or another. For very large terminologies it may be necessary to just wait this out, but this can be helped by working with local Lucene files (highly recommended), a local database, and moving load operations to a system where there are multiple processor cores and adequate memory (16GB and more). Even for loads that are memory efficient, such as those using the Spring Batch functions, indexing is still memory bound and can go much faster with more memory.