LexBIG 2008 Administrator Tips, Tricks and Gotchas

Contents

This page is intended to provide a ready-reference for LexBIG system administrators and candidate material for inclusion in the next release of the LexBIG Administration Guide.

As primary reference, see the Installation and Administration Guide in the GForge archive.

Building the LexBIG Installer from CVS

LexBIG CVS

LexBIG source folders have been distributed to the NCI CVS repository:

```
Link provided for historical purposes
cbiocvs2.nci.nih.gov:/share/content/gforge/lexbig
```

This location is maintained as a mirror for all source and packaging materials required to build LexBIG product installers. Files are refreshed from the primary development repository (maintained internally by Mayo Clinic) for each milestone release.

Development and build processes are not dependent on a particular CVS client or programming environment. However, files with the extension '.classpath' and '.project' have been inserted into each project folder to help facilitate compilation and use within the Eclipse development environment .

Folder content and purpose

- lbCVSDeploy: Contains build and packaging resources. This includes third party jar files, sample vocabularies, and scripts used to build the LexBIG installer.
- lbAdminDist: Contains source for command-line driven programs related to LexBIG administrative functions.
- lbExamplesDist: Contains source for command-line driven programs used to provide interactive examples of basic operations running against the LexBIG API.
- lbGUIDist: Contains source for the LexBIG graphical user interface.
- lbInterfacesDist: Contains source for the LexBIG API definition.
- lbImplDist: Contains source for the LexBIG API implementation.
- lgModelDist: Contains source for the LexGrid vocabulary data model.
- lbModelDist: Contains source for LexBIG extensions to the LexGrid data model.
- lbTestDist: Contains source for programs used to perform automated validation of the LexBIG runtime environment.
- lgUtilityDist: Contains source for various utility classes (e.g. persistence functions) required by the LexBIG API.

How to produce the installer

- Command line: Download and unzip the latest stable version of Ant from the Apache site . Check out all project folders from CVS, and then run the Ant command against the build.xml file provided in lbCVSDeploy. If successful, files will be created in /lbCVSDeploy/dist/.
- Eclipse: Open the build.xml file in the Eclipse Ant view and run the default task. If successful, refresh the /lbCVSDeploy folder; files will be created in /lbCVSDeploy/dist.

How to enable ydoc

- yDoc offers the capability to add UML diagrams to the generated LexBIG JavaDoc.
  If the yDoc directory is present and a valid license is located in the /yDoc/resources directory, the enhanced yDoc support is automatically enabled and used. If not, standard JavaDoc is produced.

Other notes

- Some classpath and jar dependencies have been documented for reference in file /lbCVSDeploy/ClasspathConfig.txt.
- Notes on the origin of the files are noted in /lbCVSDeploy/FILESOURCEs.txt.

JBoss Memory Settings

JBoss tends to run out of memory while running with the default settings of JBoss. To fix the problem, modify the run.conf file under JBOSS_home/bin to change the memory setting and how frequently the rmi garbage collector runs, by editing the JAVA_OPTS line as follows:

```
if [[ "x$JAVA_OPTS" = "x" ]]; then
   JAVA_OPTS="-server -XX:!MaxPermSize=128m -Xms256m -Xmx2000m -Dsun.rmi.dgc.client.gcInterval=120000
   -Dsun.rmi.dgc.server.gcInterval=120000"
fi
```

Improving Performance on MySQL/Windows

MySQL can be passed a jdbc option for the Windows local environment that may improve perfomance 30 to 50%.

Try the following values in the config.props file for the DB_URL:

```
SINGLE_DB_MODE=true
 DB_URL=jdbc:mysql:///<dbname>?socketFactory=com.mysql.jdbc.NamedPipeSocketFactory
 DB_DRIVER=org.gjt.mm.mysql.Driver
 DB_USER=root
 DB_PASSWORD=
 DB_PREFIX=lb
 DB_PARAM=
```

This uses Windows Named Pipe function and avoids use of the TCP/IP protocol. It only works when connecting with a local iteration of the MySQL database on Windows.

Error Connecting to MySQL

LexBIG is distributed with an older version of the Java MySQL driver due to licensing concerns. If LexBIG reports an error concerning establishing a connection to the MySQL server yet the MySQL CLI is able to connect, a new version of Connector/J may be required.

The latest version of Connector/J is available from MySQL.org. The new jar should be placed in the LexGrid/LexBIG/2.0.0/runtime/sqlDrivers/ directory. Remove the existing mm.mysql-2.0.6.jar to ensure that the class loader does not incorrectly load the older driver file.

Configuring Manifest Files Using Coding Scheme Manifest Entries

LexBIG Release July/Aug 2007

What is "Coding Scheme Manifest"?

A "Coding Scheme Manifest" (or simply "manifest" as used interchangeably in this document) encapsulates the user supplied values to set for a coding scheme while loading or converting an OWL (currently it is supported only for OWL type of files) 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. 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

    ```
    Historical link
    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.)

Could you please provide some example entries in a manifest file?

Here are some example coding scheme manifest entries:

```
<CodingSchemeManifestList
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:schemaLocation="http://LexGrid.org/schema/2006/01/LexOnt/CodingSchemeManifestList
  http://LexGrid.org/schema/LexBIG/2007/01/CodingSchemeManifestList.xsd"
  xmlns:owldef="http://LexGrid.org/schema/2006/01/LexOnt/CodingSchemeManifestList"
  xmlns="http://LexGrid.org/schema/2006/01/LexOnt/CodingSchemeManifestList">

      <entry id="http://www.co-ode.org/ontologies/amino-acid/2006/05/18/amino-acid.owl#">
            <codingScheme>amino</codingScheme>
            <entityDescription>This is AminoAcide Description from Manifest.</entityDescription>
            <formalName>Amino Acid Ontology</formalName>
            <defaultLanguage>en</defaultLanguage>
            <representsVersion>2005/10/11</representsVersion>
            <localName>myLocalName</localName>
            <source subRef="ontologies">COODE</source>
            <copyright>This is copyright</copyright>
            <owldef:mappings dc="mappings" xmlns="http://LexGrid.org/schema/2006/01/LexGrid/codingSchemes">
                    <supportedLanguage localId="en" urn="urn:oid:2.16.840.1.113883.6.84:en"/>
                    <supportedFormat localId="text_plain" urn="urn:oid:2.16.840.1.113883.6.10:text_plain"/>
                    <supportedSource localId="COODE" urn="http://www.co-ode.org/"/>
               </owldef:mappings>
               <associationDefinitions>
                     <assoc association="testRelation" forwardName="testRelation"
                     reverseName="testRelationInverse"
                     isTransitive="true" isReflexive="true"
                     targetCodingScheme=
                     "http://www.co-ode.org/ontologies/amino-acid/2006/05/18/outside#"/>
               </associationDefinitions>
      </entry>

      <entry id="urn:oid:2.16.840.1.113883.3.26.1.1">
            <codingScheme>NCI Thesaurus CS Manifest</codingScheme>
            <entityDescription>This is NCI Thesaurus Description from Manifest.</entityDescription>
            <formalName>NCI Thesaurus Formal From Manifest</formalName>
            <defaultLanguage>en</defaultLanguage>
            <representsVersion>manifestVersion</representsVersion>
            <localName>NCILocalName</localName>
            <source subRef="ontologies" >NCIManifestSource</source>
            <copyright>This is copyright for NCI Thesaurus</copyright> %%%
            <owldef:mappings dc="mappings" xmlns="http://LexGrid.org/schema/2006/01/LexGrid/codingSchemes">
                    <supportedSource localId="COODE" urn="http://www.co-ode.org/"/>
               </owldef:mappings>
               <associationDefinitions>
                     <assoc association="testRelation" forwardName="testRelation"
                     reverseName="testRelationInverse"
                     isTransitive="true" isReflexive="true"
                     targetCodingScheme=
                     "urn:oid:2.16.840.1.113883.3.26.1.1"/>
               </associationDefinitions>
         </entry>
</CodingSchemeManifestList>
```

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

Currently a coding scheme manifest file is only supported when loading OWL sources. Support for other formats is in-works. If you want to use the manifest file, you can supply the manifest file URI to the following methods:

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

An example code snippet:

```
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);
}
```

Generating RRF Files from UMLS

When generating source RRF files from the Metathesaurus, the UMLS Metamorphosys tool should be set to output versionless source abbreviations rather than versioned source abbreviations in any RRF subset to be loaded to LexGrid. Failing to do so will create an incomplete database leaving the association and concept tables empty.