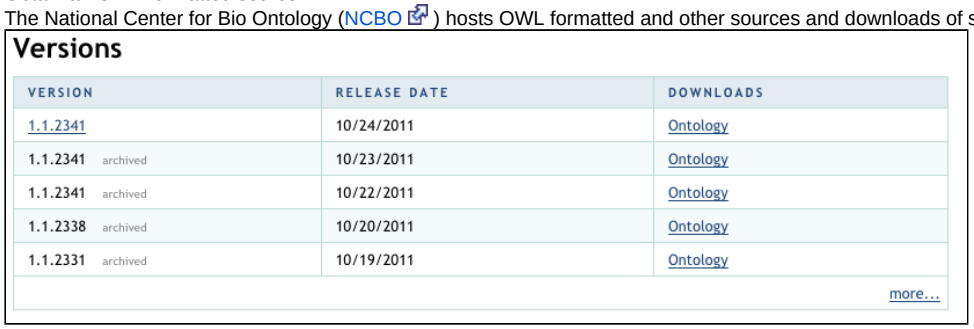
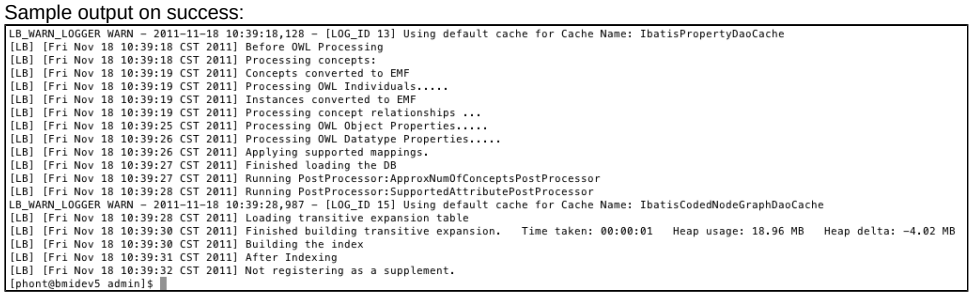
Installing OWL Terminologies in LexEVS 6.x

OWL One Terminology Loads

Most OWL formatted loads make use of the least complicated procedure to achieve an installation. However, it is a popular format and LexEVS offers some extensive options for customizing a terminology at load time. Therefore step by step instructions are provided for an OWL load.

Step	Action
1	Obtain an OWL formatted source. The National Center for Bio Ontology (NCBO) hosts OWL formatted and other sources and downloads of some sources are available.  Save the source to a local folder.

2	Load the source without options.
	Load in Linux/Unix environments:
	`./LoadOWL.sh -in "file:///home/LargeStorage/ontologies/owl/amino-acid.owl"`
	Load in Windows environments:
	`LoadOWL.bat -in "file:///home/LargeStorage/ontologies/owl/amino-acid.owl"`
	Sample output on success:
	 

Versions

VERSION	RELEASE DATE	DOWNLOADS
1.1.2341	10/24/2011	Ontology
1.1.2341 archived	10/23/2011	Ontology
1.1.2341 archived	10/22/2011	Ontology
1.1.2338 archived	10/20/2011	Ontology
1.1.2331 archived	10/19/2011	Ontology

more...

```
LB_WARN_LOGGER WARN - 2011-11-18 10:39:18,128 - [LOG_ID 13] Using default cache for Cache Name: IbatisPropertyDaoCache
[LB] [Fri Nov 18 10:39:18 CST 2011] Before OWL Processing
[LB] [Fri Nov 18 10:39:18 CST 2011] Processing concepts:
[LB] [Fri Nov 18 10:39:19 CST 2011] Concepts converted to EMF
[LB] [Fri Nov 18 10:39:19 CST 2011] Processing OWL Individuals.....
[LB] [Fri Nov 18 10:39:19 CST 2011] Instances converted to EMF
[LB] [Fri Nov 18 10:39:19 CST 2011] Processing concept relationships ...
[LB] [Fri Nov 18 10:39:25 CST 2011] Processing OWL Object Properties.....
[LB] [Fri Nov 18 10:39:26 CST 2011] Processing OWL Datatype Properties.....
[LB] [Fri Nov 18 10:39:26 CST 2011] Applying supported mappings.
[LB] [Fri Nov 18 10:39:27 CST 2011] Finished loading the DB
[LB] [Fri Nov 18 10:39:27 CST 2011] Running PostProcessor:ApproxNumOfConceptsPostProcessor
[LB] [Fri Nov 18 10:39:28 CST 2011] Running PostProcessor:SupportedAttributePostProcessor
LB_WARN_LOGGER WARN - 2011-11-18 10:39:28,987 - [LOG_ID 15] Using default cache for Cache Name: IbatisCodedNodeGraphDaoCache
[LB] [Fri Nov 18 10:39:28 CST 2011] Loading transitive expansion table
[LB] [Fri Nov 18 10:39:30 CST 2011] Finished building transitive expansion.   Time taken: 00:00:01   Heap usage: 18.96 MB   Heap delta: -4.02 MB
[LB] [Fri Nov 18 10:39:30 CST 2011] Building the index
[LB] [Fri Nov 18 10:39:31 CST 2011] After Indexing
[LB] [Fri Nov 18 10:39:32 CST 2011] Not registering as a supplement.
[phont@bmidev5 admin]$
```

3	Load source with options: Some sources may not offer much in the way of information about the terminology. Since this information is often found in resources outside the terminology itself, LexEVS offers an option to load it in an accompanying manifest file. This is an example of an OWL file with minimal metadata about the terminology: `<?xml version="1.0" encoding="UTF-8"?>` `<rdf:RDF xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"` `xmlns:rdfs="http://www.w3.org/2000/01/rdf-schema#"` `xmlns:owl="http://www.w3.org/2002/07/owl#"` `xmlns="http://www.xfront.com/owl/ontologies/camera/#"` `xmlns:camera="http://www.xfront.com/owl/ontologies/camera/#"` `xml:base="http://www.xfront.com/owl/ontologies/camera/">` `<owl:Ontology rdf:about="">` `<rdfs:comment>` `Camera OWL Ontology` `Author: Roger L. Costello` `</rdfs:comment>` `</owl:Ontology>` and this manifest file allows the user to add appropriate metadata: `<?xml version="1.0" encoding="UTF-8"?>` `<CodingSchemeManifest xsi:schemaLocation="http://LexGrid.org/schema/2010/01/LexOnt/CodingSchemeManifest http://LexGrid.org/schema/2010/01/LexOnt/CodingSchemeManifest.xsd" xmlns="http://LexGrid.org/schema/2010/01/LexOnt/CodingSchemeManifest" xmlns:lgCommon="http://LexGrid.org/schema/2010/01/LexGrid/commonTypes" xmlns:lgCon="http://LexGrid.org/schema/2010/01/LexGrid/concepts" xmlns:lgNaming="http://LexGrid.org/schema/2010/01/LexGrid/naming" xmlns:lgRel="http://LexGrid.org/schema/2010/01/LexGrid/relations" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" id="http://www.co-ode.org/ontologies/amino-acid/2006/05/18/amino-acid.owl">` `<codingScheme>Camera</codingScheme>` `<entityDescription>This is Camera Description from Manifest.</entityDescription>` `<formalName toOverride="true">CameraManifest From a Coding Scheme</formalName>` `<codingSchemeURI>CameraRegisteredName</codingSchemeURI>` `<defaultLanguage>en</defaultLanguage>` `<representsVersion>CameraV1</representsVersion>` `<localName>Camera</localName>` `<copyright>This is copyright</copyright>` `<associationDefinitions/>` `</CodingSchemeManifest>` Manifest file construction is detailed in the administrative guide. Loading in LInux/Unix `./LoadOWL.sh -in "file:///data/phont/LexEVS/test/resources/testData/camera.owl" -mf "file:///data/phont/LexEVS/test/resources/testData/Camera-manifest.xml"` Loading in Windows `LoadOWL.bat -in "file:///data/phont/LexEVS/test/resources/testData/camera.owl" -mf "file:///data/phont/LexEVS/test/resources/testData/Camera-manifest.xml"`

OWL2 Terminology Loads

Special considerations for OWL2 loads in LexEVS 6.x. OWL2 offers some unique possibilities for authoring and representation. Some terminologies contain a wide range of namespaces that need to be referenced within the code system in order for certain kinds of namespace based searches and hierarchy traversals to take place. Other options for load alterations can be made via a preferences file which can set root nodes and enable loading of Axiom based restrictions as relationships via a "strict owl" interpretation of the source.

Step	Action
1	Obtain or write an OWL2 source An example is the OBI Ontology described here: http://obi-ontology.org/page/Main_Page With source currently available here: http://svn.code.sf.net/p/obi/code/releases/2015-09-15/obi.owl

2

Create a manifest and preferences file for this source

Manifests are XML files formatted to an XML schema. They are easiest to create when using an XML editor that can reference an XML schema. The manifest schema is hosted here:

http://LexGrid.org/schema/2010/01/LexOnt/CodingSchemeManifest

For OWL2 the user can create one like the example for regular OWL above but add a tag for mappings that will help define namespaces in the OBI load:

Mappings Tag

```
    <owldef:mappings dc="mappings" xmlns="http://LexGrid.org/schema/2009/01/LexGrid/codingSchemes">
```

As tag content supply a supported coding scheme for the enclosing coding scheme OBI.

Coding Scheme

```
   <supportedCodingScheme isImported="false" localId="obi" urn="http://purl.obolibrary.org/obo/obi.
owl#" xmlns="http://LexGrid.org/schema/2008/01/LexGrid/codingSchemes"/>
```

After this tag define the namespace for the enclosing coding scheme with a reference to the enclosing coding scheme in the equivalent coding scheme attribute

Enclosing Scheme Namespace

```
 <supportedNamespace localId="obi" uri="http://purl.obolibrary.org/obo/obi.owl#"
equivalentCodingScheme="obi">obi</supportedNamespace>
```

All other namespaces can be defined in the same way with references from the original OWL source which are found in the xmlns attributes of the "<rdf:RDF tag." Each with a reference to the enclosing coding scheme using the equivalentCodingScheme attribute

Supported Namespace

```
        <supportedNamespace localId="dc" uri="http://purl.org/dc/elements/1.1/"
equivalentCodingScheme="obi">dc</supportedNamespace>
        <supportedNamespace localId="doap" uri="http://usefulinc.com/ns/doap#" equivalentCodingScheme="
obi">doap</supportedNamespace>
        <supportedNamespace localId="foaf" uri="http://xmlns.com/foaf/0.1/" equivalentCodingScheme="
obi">foaf</supportedNamespace>
        <supportedNamespace localId="obo" uri="http://purl.obolibrary.org/obo/"
equivalentCodingScheme="obi">obo</supportedNamespace>
        <supportedNamespace localId="oboInOwl" uri="http://www.geneontology.org/formats/oboInOwl#"
equivalentCodingScheme="obi">oboInOwl</supportedNamespace>
        <supportedNamespace localId="owl" uri="http://www.w3.org/2002/07/owl" equivalentCodingScheme="
obi">owl</supportedNamespace>
        <supportedNamespace localId="protege" uri="http://protege.stanford.edu/plugins/owl/protege#"
equivalentCodingScheme="obi">protege</supportedNamespace>
        <supportedNamespace localId="rdf" uri="http://www.w3.org/1999/02/22-rdf-syntax-ns"
equivalentCodingScheme="obi">rdf</supportedNamespace>
        <supportedNamespace localId="rdfs" uri="http://www.w3.org/2000/01/rdf-schema"
equivalentCodingScheme="obi">rdfs</supportedNamespace>
        <supportedNamespace localId="xsd" uri="http://www.w3.org/2001/XMLSchema"
equivalentCodingScheme="obi">xsd</supportedNamespace>
        <supportedNamespace localId="xml" uri="http://www.w3.org/XML/1998/namespace"
equivalentCodingScheme="obi">xml</supportedNamespace>
```

Attached is a full example of a working manifest for OBI

OBI_Apr2015_With_NamespaceToCS_MF.xml

Preferences files work in a similar way to the manifest. Instead of updating coding scheme data, they initiate programmatic switches determining how a variety of source definitions can be loaded. The preferences XML scheme is defined here:

http://LexGrid.org/schema/LexBIG/2009/01/Preferences/load/OWLLoadPreferences

	This includes setting nodes designated as root, i.e. without relational parent codes of any kind.
	
	Match Root Name
	
	`<MatchRootName>(BFO_0000001|Project|SVNRepository|ObsoleteClass|Version)</MatchRootName>`
	
	Other applications for OWL2 include loading Axiom based restrictions as a variety of relationships. This is accomplished by setting the strict owl tag value to true:
	
	Strict OWL
	
	`<StrictOWLImplementation>true</StrictOWLImplementation>`
	
	A full working preferences file is attached below
	
	OBI_StrictOWL_PF.xml

3	Load source with options using option files `./LoadOWL2.sh -in "file:///data/phont/LexEVS/test/resources/testData/obi.owl" -mf "file:///data/phont/LexEVS/test/resources/testData/OBI_Apr2015_With_NamespaceToCS_MF.xml" -lp "file:///data/phont/LexEVS/test/resources/testData/OBI_StrictOWL_PF.xml"`