4 - Advanced LexEVS 6.x Vocabulary Administration

Contents of this Page

LexEVS Administration Links

Introduction

This document is a section of the Administration Guide.

Using a Manifest to Change Vocabulary MetaData

LexEVS manages vocabulary meta data updates during load time or post load using optional manifest loads.

What is a Coding Scheme Manifest?

A "Coding Scheme Manifest" (or manifest) allows the user to set meta data values for a coding scheme. This can be done while loading or converting a LexGrid "XML", "NCI MetaThesaurus", "NCI OWL","OWL", "OBO", "UMLS RRF File", or"HL7 RIM Database" source to LexGrid format or post load.

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):

id

- 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://130.88.198.11/co-ode-files/ontologies/amino-acid.owl . This string is also set as the coding scheme's registered name field in the LexGrid model.

codingScheme

- 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.

entityDescription

- 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.

formalName

- 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.

codingSchemeURI

- Coding Scheme Manifest entry field: **codingSchemeURI**
  - Type: lgCommon:URI
  - Required: No
  - "To Add" flag set: Yes
  - Description: This value will be set for 'coding scheme URI' 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.

defaultLanguage

- 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.

representsVersion

- 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.

localName

- 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.

source

- 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.

copyright

- 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.

mappings

- 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.

associationDefinitions

- 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:

Java 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-acid.owl").toURI(),
                new File("resources/testData/aa-manifest.xml").toURI(),true,true);
}
```

For all other manifest loads the following methods are employed.

Java Code Snippet

```
    // 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);
```

Sample Coding Scheme Manifest for the NCI Thesaurus

```
 <CodingSchemeManifest xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
     xsi:schemaLocation="http://LexGrid.org/schema/2009/01/LexOnt/CodingSchemeManifest http://LexGrid.org/schema
/2009/01/LexOnt/CodingSchemeManifest.xsd"
     xmlns:owldef="http://LexGrid.org/schema/2009/01/LexOnt/CodingSchemeManifest"
     xmlns:lgNaming="http://LexGrid.org/schema/2009/01/LexGrid/naming"
     xmlns:lgRel="http://LexGrid.org/schema/2009/01/LexGrid/relations"
     xmlns="http://LexGrid.org/schema/2009/01/LexOnt/CodingSchemeManifest"
     id="http://ncicb.nci.nih.gov/xml/owl/EVS/Thesaurus.owl#">
    <codingScheme toOverride="true">NCI_Thesaurus</codingScheme>
    <entityDescription toOverride="true">NCI Thesaurus</entityDescription>
    <formalName toOverride="true">NCI Thesaurus</formalName>
    <codingSchemeURI>http://ncicb.nci.nih.gov/xml/owl/EVS/Thesaurus.owl#</codingSchemeURI>
    <defaultLanguage toOverride="true">en</defaultLanguage>
    <representsVersion toOverride="true">09.06b</representsVersion>
    <localName toAdd="true">NCI Thesaurus</localName>
    <localName toAdd="true">NCI_Thesaurus</localName>
    <owldef:mappings xmlns="http://LexGrid.org/schema/2009/01/LexGrid/codingSchemes">
        <lgNaming:supportedCodingScheme localId="NCI_Thesaurus" uri="http://ncicb.nci.nih.gov/xml/owl/EVS
/Thesaurus.owl#" isImported="true" />
        <lgNaming:supportedRepresentationalForm localId="text_plain" uri="urn:oid:2.16.840.1.113883.6.10:
text_plain" />
    </owldef:mappings>
      <owldef:associationDefinitions toUpdate="true">
        <assoc associationName="Has_Salt_Form" entityCode="A5" forwardName="Has_Salt_Form" reverseName="
Has_Free_Acid_Or_Base_Form" />
    </owldef:associationDefinitions>
</CodingSchemeManifest>
```

Dealing with Large Terminologies

Loading Large RRF Terminologies (Examples: (NCI Metathesaurus, SNOMEDCT, LOINC, etc):

- Primary Key Strategy - see (DB_PRIMARY_KEY_STRATEGY Config Setting)
  - Sequential Integer Primary Key (SEQUENTIAL_INTEGER) is the best strategy for large loads. This allows the database to insert records into the index in a sequential manner, which is more efficient. If GUID strategy is used, records will be inserted into the index tree at random locations, resulting in index re-balancing after every insert.
- Hardware is very important to large content loads.
  - RRF Loads are loaded in a multi-threaded manner. Multi-processor servers will give the best performance.
  - If possible, seperate the database server and the loader server.
- Monitoring a load
  - Monitor all LexEVS logs (both 'load' and 'full' log).
  - If using MySQL, use INNODB tools to monitor Inserts per second. (SHOW INNODB STATUS)

Load Time Preferences

Preferences for loading elements of sources such as OWL can be done at load time.

General Meta Data File Association Preferences

This value can be adjusted by creating an XML file that resolves against this schema: http://LexGrid.org/schema/LexBIG/2009/01/Preferences/load/LoadPreferences

XMLMetadataFilePath

Any xml document can be assigned as metadata to a newly loaded coding scheme. The xml document is broken down into individual tags and values, which are then searchable through the LexBIG Service Metadata interface. This parameter indicates the path of xml metadata assigned during the current load operation. For most loaders, the given path serves strictly as an option to input user-specified data. For The NCI Metathesaurus loader, metadata is automatically generated and assigned to the coding scheme. In these cases, the generated xml will be output to the given file, overwriting any existing content.

Owl Loader Preferences

These values can be adjusted by creating an XML file that resolves against this schema: http://LexGrid.org/schema/LexBIG/2009/01/Preferences/load/OWLLoadPreferences

PropnamePrimitive

Entities can be assigned a property that indicates whether or not it is considered primitive (having no equivalent classes). This preference controls the name of the property that is created; the property value will indicate true or false. If not specified, the name 'primitive' is assumed.

PropnameType

Anonymous OWL classes of type OWLNAryLogicalClass can be assigned properties that indicate the nature or type of component logical operations. This preference controls the name of the property that is created; the property value will indicate the logical operation (e.g. owl:oneOf). If not specified, the name 'type' is assumed.

MatchConceptCode

This preference allows for entity codes to be derived from a specific RDF property. The provided string is interpreted as a regular expression to be compared against properties assigned to each processed class. If a property name matches the regular expression, the property value is assigned as the entity code. If not specified no default match is assumed, and the entity code is derived from the RDF resource name.

MatchConceptStatus

This preference allows for entity status to be derived from a specific RDF property. The provided string is interpreted as a regular expression to be compared against properties assigned to each processed class. If a property name matches the regular expression, the property value is assigned as the entity status. If not specified, the regular expression of ('concept_status') is assumed, and if not matched no status string is assigned (the isActive boolean flag will still be set based on deprecation).

MatchNoopNamespaces

This preference allows for classes to be selectively ignored on import to LexGrid. The provided string is interpreted as a regular expression to be compared against class namespace. If matched, a counterpart entity is not created in the LexGrid coding scheme. If not provided, the expression

```
'(:|@_:|protege:|xsp:).*'
```

is assumed.

MatchRootName

This preference allows for custom declaration of root concepts for hierarchical relationships. The provided string is interpreted as a regular expression to be compared against the resource name for each class. If matched, the node is designated as a root in the supported hierarchy metadata. If not specified, root nodes are identified by having a superclass of owl:thing.

MatchXMLRepformNames

If processing of complex properties is enabled (see ProcessComplexProps preference), this preference allows for identification of representational form names contained by XML fragments embedded within rdf property text. The provided string is interpreted as a regular expression and compared against the XML tags in each fragment. If not specified, the default expression '(term-group)' is assumed.

MatchXMLSourceNames

If processing of complex properties is enabled (see ProcessComplexProps preference), this preference allows for identification of source names contained by XML fragments embedded within rdf property text. The provided string is interpreted as a regular expression and compared against the XML tags in each fragment. If not specified, the default expression '(term-source|def-source)' is assumed.

MatchXMLTextNames

If processing of complex properties is enabled (see ProcessComplexProps preference), this preference allows for identification of descriptive text contained by XML fragments embedded within rdf property text. The provided string is interpreted as a regular expression and compared against the XML tags in each fragment. If not specified, the default expression '(term-name|def-definition|go-term)' is assumed.

IsDBXrefSource

If processing of complex properties is enabled (see ProcessComplexProps preference) and source, this preference allows for identification of ISBN cross reference information in xml element text. If not specified, the default of 'false' is assumed.

IsDBXrefRepform

If processing of complex properties is enabled (see ProcessComplexProps preference) and source, this preference allows for identification of representational form cross reference information in xml element text. If not specified, the default of 'false' is assumed.

ProcessComplexProps

Indicates whether rdf property text will be evaluated to detect and process embedded XML. This is a master switch controlling whether the MatchXML* and isDBXRef* preferences are acknowledged. The default is false.

StrictOWLImplementation

Controls the relationship between restrictions of anonymous classes and parent concepts. If true, restrictions for anonymous classes are not connected to the parent concepts. The default is true.

CreateConceptForObjectProp

Controls whether concept entities are created for object properties defined in the OWL source. The default is false.

DatatypePropSwitch

Controls how data type properties are converted to components of the LexGrid model. If 'association' is specified, each data type property is recorded in LexGrid as an entity-to-entity relationship. If 'conceptProperty' is specified, traditional LexGrid properties are created and assigned directly to new entities . If 'both' is specified, both entity relationships and standard LexGrid entity properties are generated. The default is 'both'.

PrioritizedCommentNames

Indicates a list of rdf property names to be attributed special semantic significance as comments in the LexGrid model. If not specified, the default if 'DesignNote, Editor_Note, Citation, and VA_Workflow_Comment' is assumed. If not matched, only generic properties are created.

PrioritizedDefinitionNames

Indicates a list of rdf property names to be attributed special semantic significance as definitions in the LexGrid model. If not specified, the default of 'DEFINITION", dDEFINITION, LONG_DEFINITION, ALT_DEFINITION, ALT_LONG_DEFINITION, and MeSH_Definition' is assumed.

PrioritizedPresentationNames

Indicates a list of rdf property names to be attributed special semantic significance as definitions in the LexGrid model. If not specified, the default of 'NCI_Preferred_Term, Preferred_Name, Display_Name, Search_Name, FULL_SYN, Synonym, VA_Print_Name, VA_National_Formulary_Name, VA_Abbreviation, VA_Dose_Form_Print_Name, VA_Trade_Name, MeSH_Name, NDFRT_Name, RxNorm_Name' is assumed.

UMLS SEMNET Preferences

This value can be adjusted by creating an XML file that resolves against this schema: http://LexGrid.org/schema/LexBIG/2009/01/Preferences/load/SemNetLoadPreferences

SemNetLoaderPreferences

The load parameter controls which inherited relationships are loaded and navigable within LexBIG. When selecting the option not to load inherited relationships, all associations are extracted from the source file SRSTR (stated relations). When loading all inherited relations, associations are extracted from the source file SRSTRE1 (classified relations).

At NCI's request we provided an additional option to load only stated relations for direct subclass ('is_a') associations, but inherited relationships for all other associations. This was intended to provide similar behavior to their provision of OWL sources.

> **Note**
>
> Direct or stated relationships are always imported, regardless of the selected option.

- value="0" Only load direct or stated relationships.
- value="1" Load all inherited relationships.
- value="2" Load all inherited relationships except is_a.

Applying Revisions to a Coding Scheme

A Revision Overview

CodingSchemes can be extensively revised by loading a Revision object in LexGrid XML format. A coding scheme Revision can be created to resolve against a "revision" schema URL and loaded to a coding scheme current in the service. This revision is tracked within the service history. Revision function centers around LexGrid model elements that inherit from the Versionable element. Versionable classes and attributes include those "types" of Versionable and any attributes inherited from this element. Whenever a Versionable element appears in a revision it is accompanied by an EntryState el ement which helps define it's role in the revision process.

A Revision Example

For instance, the following revision defines a new association for the coding scheme AutomobilesAD. The AssociationTarget class is a Versionable type, but the AssociationSource is not. So the AssociationTarget revision is defined by an EntryState element with a changeType value "NEW".

```
<revision xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xsi:schemaLocation="http://LexGrid.org/schema/2010/01/LexGrid/versions http://LexGrid.org/schema/2010/01
/LexGrid/versions.xsd"
    xmlns="http://LexGrid.org/schema/2010/01/LexGrid/versions"
    xmlns:lgBuiltin="http://LexGrid.org/schema/2010/01/LexGrid/builtins"
    xmlns:lgCommon="http://LexGrid.org/schema/2010/01/LexGrid/commonTypes"
    xmlns:lgCon="http://LexGrid.org/schema/2010/01/LexGrid/concepts"
    xmlns:lgRel="http://LexGrid.org/schema/2010/01/LexGrid/relations"
    xmlns:lgCS="http://LexGrid.org/schema/2010/01/LexGrid/codingSchemes"
    xmlns:lgNaming="http://LexGrid.org/schema/2010/01/LexGrid/naming"
    xmlns:lgVD="http://LexGrid.org/schema/2010/01/LexGrid/valueDomains"
    xmlns:NCIHistory="http://LexGrid.org/schema/2010/01/LexGrid/NCIHistory" revisionId="
testRelease2010Jan_testData">
    <changedEntry>
        <changedCodingSchemeEntry codingSchemeName="AutomobilesAD"
            codingSchemeURI="urn:oid:22.22.0.2" representsVersion="2.0">
            <lgCommon:entryState containingRevision="testRelease2010Jan_testData"
                relativeOrder="0" changeType="DEPENDENT"/>
            <lgCS:mappings/>
            <lgCS:relations containerName="rel">
                <lgCommon:entryState containingRevision="testRelease2010Jan_testData"
                    relativeOrder="0" changeType="DEPENDENT"/>
                <lgRel:associationPredicate associationName="sameAs">
                    <lgRel:source sourceEntityCode="005"
                        sourceEntityCodeNamespace="AutomobilesAD">
                        <lgRel:targetData associationInstanceId="inst00b" isActive="true"
                            status="high" isInferred="false" isDefining="false"
                            effectiveDate="2000-12-01T01:29:35"
                            expirationDate="2019-10-01T01:45:33">
                            <lgCommon:owner>BSI</lgCommon:owner>
                            <lgCommon:entryState
                                containingRevision="testRelease2010Jan_testData"
                                relativeOrder="0" changeType="NEW"/>
                            <lgRel:usageContext>contextText001</lgRel:usageContext>
                            <lgRel:associationQualification associationQualifier="qual001">
                                <lgRel:qualifierText>qualTextValue001</lgRel:qualifierText>
                            </lgRel:associationQualification>
                            <lgRel:associationDataText>testData001</lgRel:associationDataText>
                        </lgRel:targetData>
                    </lgRel:source>
                </lgRel:associationPredicate>
            </lgCS:relations>
        </changedCodingSchemeEntry>
    </changedEntry>
</revision>
```

Notice that even though an AssociationSource contains a collection of targets, it is not a Versionable element itself, so the revision definition for an association is in the association target. A collection of sources is contained in another unversioned element the AssociationPredicate . The predicate's container, Relations is a Versionable element but it is already defined in the coding scheme so it is defined as a "DEPENDENT" revision. Similarly the containing CodingScheme itself is a Versionable element also defined as a "DEPENDENT" revision. Notice the revisionId attribute of the the top level Revision element and how it corresponds to the containingRevision attribute value on all the EntryState elements. This value must differ from the current revisionId of the coding scheme being revised.

Change Type Definitions

There are 5 changeType definitions. Each change type needs to be applied in it's own context as follows:

- NEW - to create a new versionable element
- MODIFY - to change the attributes of an existing versionable element
- VERSIONABLE - Versionable attribute has changed since prior version. Versionable attributes include: isActive, status, owner, effectiveDate or expirationDate.
- DEPENDENT - The status of a dependent entry has changed since the last version. Dependent entities include properties, codedEntries for codingSchemes, associationInstances, etc.
- REMOVE - Versionable entry was removed as of this version. This is not the same as deprecated, as the entity ceases to exist in future versions.

Revision Resources

Revisions in LexGrid are discussed in more detail here:

- Authoring Design Document

Post Processing Options

Post load processing algorithms allow users to access information about the source that may only be available post load and apply to coding scheme meta-data.

Post Processor Extensions Overview

Ontology Format Adding Post Processor

This post processor adds a property to the coding scheme indicating it's source format. This formatting note is detected by the LexEVS RDF exporter which tailors LexGrid to RDF mapping based on the original format of source of the loaded coding scheme.

Approximate Number Of Concepts Post Processor

This post processor tracks concept numbers in the coding scheme and applies them to the approxNumOfConcepts attribute of the CodingScheme.

Supported Attribute Post Processor

Determines and loads supported attributes for the following:

- Supported Coding Scheme for the loaded coding scheme
- Supported Formats
- Supported Languages
- Supported EntityTypes
- Supported Properties
- Supported Associations

Post Processor Example Code

```
                AbsoluteCodingSchemeVersionReference ref = new AbsoluteCodingSchemeVersionReference();
                       ref.setCodingSchemeURN("urn:oid:11.00.11.1");
                ref.setCodingSchemeVersion("1.0");
                LexBIGService lbs = lb_gui_.getLbs();

                try {
                    LoaderPostProcessor postProcessor =
                        lbs.getServiceManager(null).getExtensionRegistry().
                        getGenericExtension("SupportedAttributePostProcessor", LoaderPostProcessor.class);

                    LoaderPostProcessRunner loaderPostProcessRunner = new LoaderPostProcessRunner
(postProcessor);
                    loaderPostProcessRunner.runProcess(ref, null));
```

Post Processor Post Load Application in lbGUI

Run Post Processor from the GUI

Coding Scheme Supplements

Supplements to a given coding scheme can be loaded as full coding schemes. Vocabularies can be augmented in LexEVS using the coding scheme supplement API. This effectively unions one code set with another in LexEVS.

Supplement Loading Scenario

In one possible scenario a user could add entities to an existing coding scheme by creating a new coding scheme in LexGrid XML format and load the new scheme as a supplement to the existing scheme.

Original Coding Scheme Excerpt

```
<!--
    This file was created using LexGrid.  You can find out more about LexGrid at http://informatics.mayo.edu/ .
    Generated at: 12/14/06 3:38 PM
    Generated by: org.LexGrid.emf
-->
<codingScheme xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:schemaLocation="http://LexGrid.org/schema/2010/01/LexGrid/codingSchemes  https://ncisvn.nci.nih.gov/svn
/lexevs/base/v6/trunk/lexgrid_model/lgModel/master/codingSchemes.xsd"
  xmlns="http://LexGrid.org/schema/2010/01/LexGrid/codingSchemes"
  xmlns:lgBuiltin="http://LexGrid.org/schema/2010/01/LexGrid/builtins"
  xmlns:lgCommon="http://LexGrid.org/schema/2010/01/LexGrid/commonTypes"
  xmlns:lgCon="http://LexGrid.org/schema/2010/01/LexGrid/concepts"
  xmlns:lgRel="http://LexGrid.org/schema/2010/01/LexGrid/relations"
  xmlns:lgCS="http://LexGrid.org/schema/2010/01/LexGrid/codingSchemes"
  xmlns:lgLDAP="http://LexGrid.org/schema/2010/01/LexGrid/ldap"
  xmlns:lgNaming="http://LexGrid.org/schema/2010/01/LexGrid/naming"
  xmlns:lgService="http://LexGrid.org/schema/2010/01/LexGrid/service"
  xmlns:lgVD="http://LexGrid.org/schema/2010/01/LexGrid/valueDomains"
  xmlns:lgVer="http://LexGrid.org/schema/2010/01/LexGrid/versions"
  xmlns:NCIHistory="http://LexGrid.org/schema/2010/01/LexGrid/NCIHistory"
  approxNumConcepts="5" codingSchemeName="Automobiles" defaultLanguage="en" formalName="autos" codingSchemeURI="
urn:oid:11.11.0.1" representsVersion="1.0">
  <lgCommon:entityDescription>Automobiles</lgCommon:entityDescription>
  <localName>11.11.0.1</localName>
  <localName>Automobiles</localName>
  <localName>SomeOtherValue</localName>
  <copyright>Copyright by Mayo Clinic.</copyright>

  <mappings>
    <lgNaming:supportedAssociation localId="hasSubtype" uri="urn:oid:1.3.6.1.4.1.2114.108.1.8.1">hasSubtype<
/lgNaming:supportedAssociation>
    <lgNaming:supportedAssociation localId="uses" uri="urn:oid:11.11.0.1">uses</lgNaming:supportedAssociation>
    <lgNaming:supportedAssociation localId="A1" uri="http://A1.org" entityCode="AssocEntity"
entityCodeNamespace="Automobiles" codingScheme="Automobiles">A1</lgNaming:supportedAssociation>
    <lgNaming:supportedAssociationQualifier localId="hasEngine" uri="www.something.com">hasEngine</lgNaming:
supportedAssociationQualifier>
    <lgNaming:supportedAssociationQualifier localId="since" uri="www.since.com">since</lgNaming:
supportedAssociationQualifier>
    <lgNaming:supportedAssociationQualifier localId="sold" uri="www.sold.com">sold</lgNaming:
supportedAssociationQualifier>
    <lgNaming:supportedCodingScheme localId="Automobiles" uri="urn:oid:11.11.0.1">Automobiles</lgNaming:
supportedCodingScheme>
    <lgNaming:supportedCodingScheme localId="ExpendableParts" uri="urn:oid:11.11.0.50">Expendable Parts<
/lgNaming:supportedCodingScheme>
    <lgNaming:supportedCodingScheme localId="GermanMadeParts" uri="urn:oid:11.11.0.2">German Made Parts<
/lgNaming:supportedCodingScheme>
    <lgNaming:supportedContainerName localId="relations">relations</lgNaming:supportedContainerName>
    <lgNaming:supportedDataType localId="testhtml">test/html</lgNaming:supportedDataType>
    <lgNaming:supportedDataType localId="textplain">text/plain</lgNaming:supportedDataType>
    <lgNaming:supportedHierarchy localId="is_a" associationNames="hasSubtype" isForwardNavigable="true"
rootCode="@">hasSubtype</lgNaming:supportedHierarchy>
    <lgNaming:supportedLanguage localId="en" uri="www.en.org/orsomething">en</lgNaming:supportedLanguage>
    <lgNaming:supportedNamespace localId="Automobiles" uri="urn:oid:11.11.0.1" equivalentCodingScheme="
Automobiles">Automobiles</lgNaming:supportedNamespace>
    <lgNaming:supportedNamespace localId="ExpendableParts" uri="urn:oid:11.11.0.50" equivalentCodingScheme="
ExpendableParts">Expendable Parts</lgNaming:supportedNamespace>
    <lgNaming:supportedNamespace localId="GermanMadePartsNamespace" uri="urn:oid:11.11.0.2"
equivalentCodingScheme="GermanMadeParts">German Made Parts</lgNaming:supportedNamespace>
    <lgNaming:supportedNamespace localId="TestForSameCodeNamespace" uri="urn:oid:11.11.0.99"
>TestForSameCodeNamespace</lgNaming:supportedNamespace>
    <lgNaming:supportedProperty localId="definition">definition</lgNaming:supportedProperty>
    <lgNaming:supportedProperty localId="textualPresentation" propertyType="presentation">textualPresentation<
/lgNaming:supportedProperty>
```

```
    <lgNaming:supportedProperty localId="genericProperty" >genericProperty</lgNaming:supportedProperty>
    <lgNaming:supportedSource localId="lexgrid.org">lexgrid.org</lgNaming:supportedSource>
    <lgNaming:supportedSource localId="_111101">11.11.0.1</lgNaming:supportedSource>
  </mappings>
  <properties>
    <lgCommon:property expirationDate="2001-12-17T09:30:47Z" language="en" propertyType="property" status="
sampleStatus" propertyId="p1" effectiveDate="2001-12-17T09:30:47Z" isActive="true" propertyName="
codingSchemeProp">
      <lgCommon:owner >sampleOwner</lgCommon:owner>
      <lgCommon:source role="sampleRole" subRef="sampleSubRef">lexgrid.org</lgCommon:source>
      <lgCommon:usageContext>sampleUsageContext</lgCommon:usageContext>
      <lgCommon:propertyQualifier propertyQualifierName="samplePropertyQualifier">
        <lgCommon:value>Property Qualifier Text</lgCommon:value>
      </lgCommon:propertyQualifier>
      <lgCommon:value>Property Text</lgCommon:value>
    </lgCommon:property>
</properties>
  <entities>
    <lgCon:entity entityCode="005" entityCodeNamespace="Automobiles" isActive="true">
      <lgCommon:entityDescription>Domestic Auto Makers</lgCommon:entityDescription>
      <lgCon:entityType>concept</lgCon:entityType>
      <lgCon:presentation propertyName="textualPresentation" propertyId="p1" isPreferred="true">
              <lgCommon:source role="sampleSource" subRef="sampleSubRef1">lexgrid.org</lgCommon:source>
        <lgCommon:source role="sampleSource" subRef="sampleSubRef2">lexgrid.org</lgCommon:source>
        <lgCommon:value>Domestic Auto Makers</lgCommon:value>
      </lgCon:presentation>
      <lgCon:presentation propertyName="textualPresentation" propertyId="p2" isPreferred="false">
        <lgCommon:value>American Car Companies</lgCommon:value>
      </lgCon:presentation>
    </lgCon:entity>
```

Full Coding Scheme to be Loaded as Extension

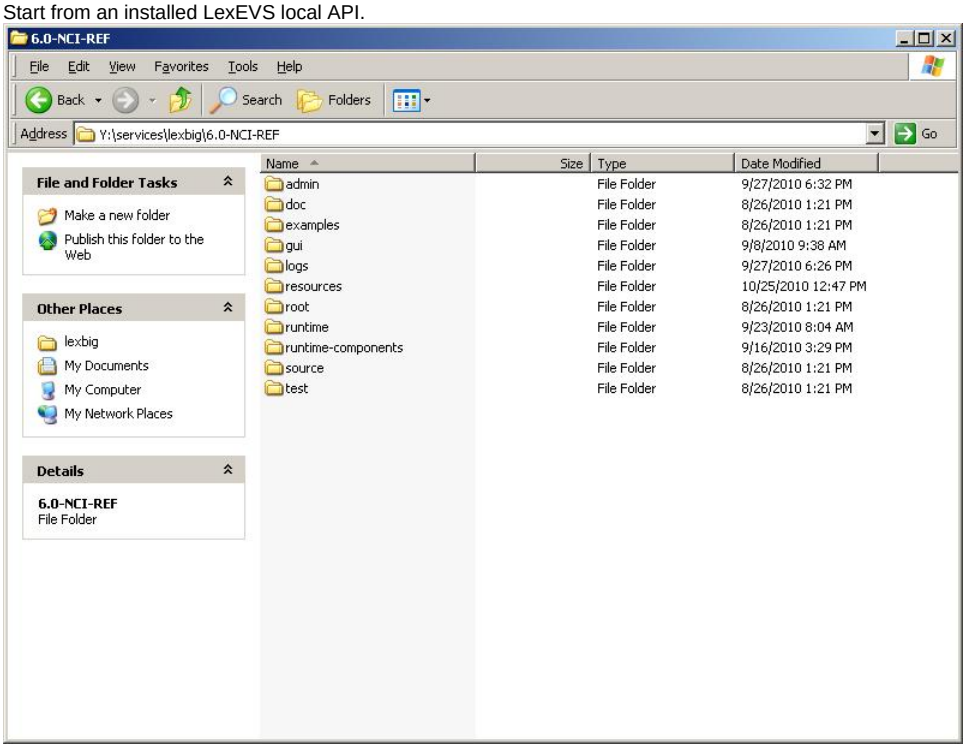
```
<lgCS:codingScheme
    xmlns:lgBuiltin="http://LexGrid.org/schema/2010/01/LexGrid/builtins"
    xmlns:lgCommon="http://LexGrid.org/schema/2010/01/LexGrid/commonTypes"
    xmlns:lgCon="http://LexGrid.org/schema/2010/01/LexGrid/concepts"
    xmlns:lgCS="http://LexGrid.org/schema/2010/01/LexGrid/codingSchemes"
    xmlns:lgNaming="http://LexGrid.org/schema/2010/01/LexGrid/naming"
    xmlns:lgRel="http://LexGrid.org/schema/2010/01/LexGrid/relations"
    xmlns:lgVD="http://LexGrid.org/schema/2010/01/LexGrid/valueSets"
    xmlns:lgVer="http://LexGrid.org/schema/2010/01/LexGrid/versions"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xsi:schemaLocation="http://LexGrid.org/schema/2010/01/LexGrid/codingSchemes  http://LexGrid.org/schema/2010
/01/LexGrid/codingSchemes.xsd"
    codingSchemeName="Automobiles_extension"
    codingSchemeURI="urn:oid:11.11.0.1.1-extension"
    formalName="Automobiles Extension" defaultLanguage="en"
    approxNumConcepts="1" representsVersion="1.0-extension">

    <lgCommon:entityDescription>Automobiles Extension</lgCommon:entityDescription>
    <lgCS:mappings>
    <lgNaming:supportedAssociation localId="hasSubtype"  entityCode="hasSubtype" entityCodeNamespace="
Automobiles" codingScheme="Automobiles">hasSubtype</lgNaming:supportedAssociation>
    <lgNaming:supportedCodingScheme localId="Automobiles_extension" uri="urn:oid:11.11.0.1-extension"
>Automobiles</lgNaming:supportedCodingScheme>
    <lgNaming:supportedCodingScheme localId="Automobiles" uri="urn:oid:11.11.0.1">Automobiles</lgNaming:
supportedCodingScheme>
    <lgNaming:supportedHierarchy localId="is_a" associationNames="hasSubtype" isForwardNavigable="true"
rootCode="@">hasSubtype</lgNaming:supportedHierarchy>
    <lgNaming:supportedNamespace localId="Automobiles" uri="urn:oid:11.11.0.1" equivalentCodingScheme="
Automobiles">Automobiles</lgNaming:supportedNamespace>
    <lgNaming:supportedNamespace localId="Automobiles_extension" uri="urn:oid:11.11.0.1.1"
equivalentCodingScheme="Automobiles_extension">Automobiles_extension</lgNaming:supportedNamespace>


    </lgCS:mappings>

    <lgCS:properties/>
```

```
    <lgCS:entities>
        <lgCon:entity
            xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
            xsi:schemaLocation="http://LexGrid.org/schema/2010/01/LexGrid/codingSchemes  http://LexGrid.org
/schema/2010/01/LexGrid/codingSchemes.xsd"
            isActive="true" entityCode="Cadillac"
            entityCodeNamespace="Automobiles_extension" isAnonymous="false" isDefined="false" >

           <lgCommon:entityDescription>Cadillac</lgCommon:entityDescription>
           <lgCon:entityType>concept</lgCon:entityType>
           <lgCon:presentation propertyName="textualPresentation" propertyId="p1" isPreferred="true">
                                <lgCommon:source>Extension</lgCommon:source>
                                <lgCommon:value dataType="string">Cadillac</lgCommon:value>
           </lgCon:presentation>
        </lgCon:entity>
        <lgCon:entity
            xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
            xsi:schemaLocation="http://LexGrid.org/schema/2010/01/LexGrid/codingSchemes  http://LexGrid.org
/schema/2010/01/LexGrid/codingSchemes.xsd"
            isActive="true" entityCode="DeVille"
            entityCodeNamespace="Automobiles_extension" isAnonymous="false" isDefined="false" >

           <lgCommon:entityDescription>DeVille</lgCommon:entityDescription>
           <lgCon:entityType>concept</lgCon:entityType>
           <lgCon:presentation propertyName="textualPresentation" propertyId="p1" isPreferred="true">
                                <lgCommon:source>Extension</lgCommon:source>
                                <lgCommon:value dataType="string">DeVille</lgCommon:value>
           </lgCon:presentation>
        </lgCon:entity>
    </lgCS:entities>

    <lgCS:relations containerName="relations">
        <lgRel:associationPredicate  associationName="hasSubtype">
                        <lgRel:source sourceEntityCodeNamespace="Automobiles" sourceEntityCode="GM">
                        <lgRel:target targetEntityCodeNamespace="Automobiles_extension" targetEntityCode="
Cadillac"/>
            </lgRel:source>
            <lgRel:source sourceEntityCodeNamespace="Automobiles_extension" sourceEntityCode="Cadillac">
                        <lgRel:target targetEntityCodeNamespace="Automobiles_extension" targetEntityCode="
DeVille"/>
            </lgRel:source>
 </lgRel:associationPredicate>
    </lgCS:relations>
</lgCS:codingScheme>
```

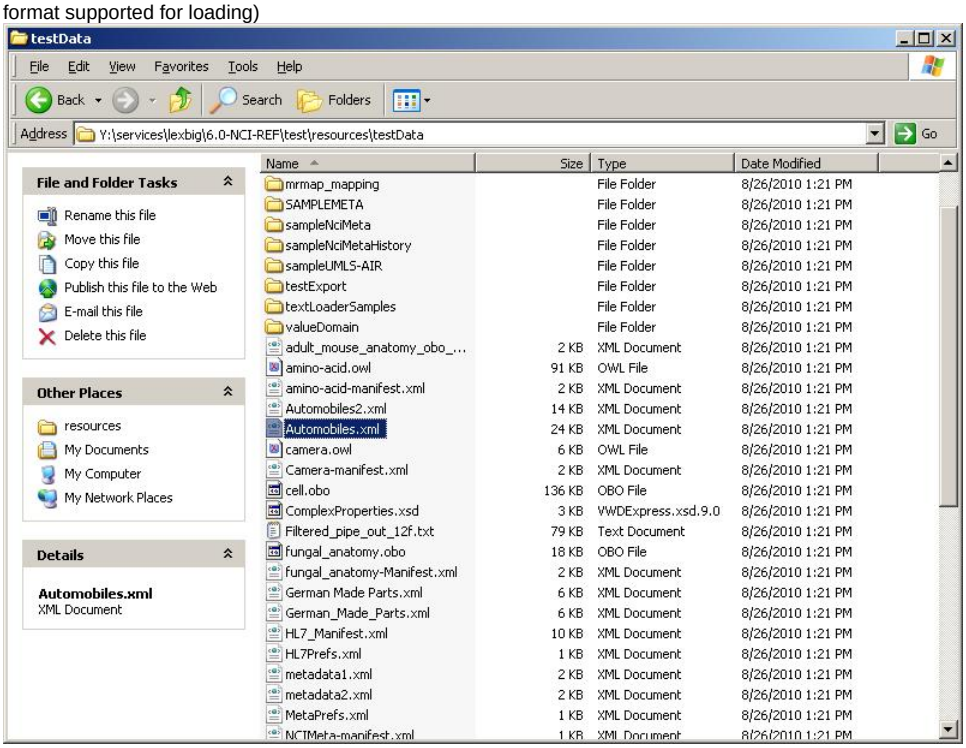
Extension Loading Scenario

Users interested in this functionality might try the following to see it in action:

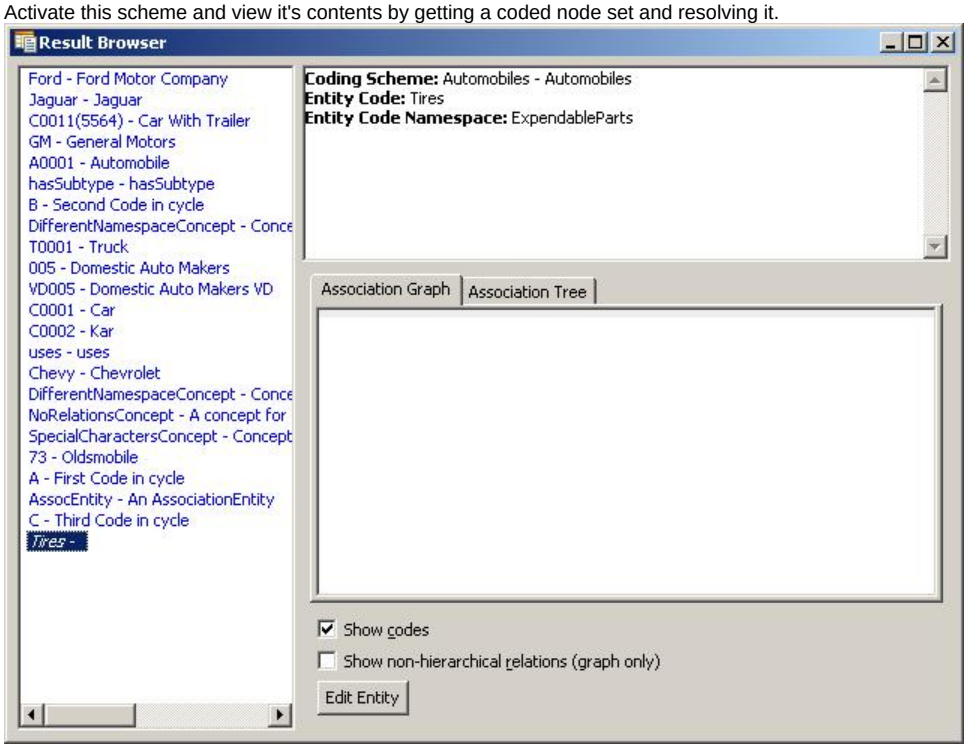
1. Start from an installed LexEVS local API.



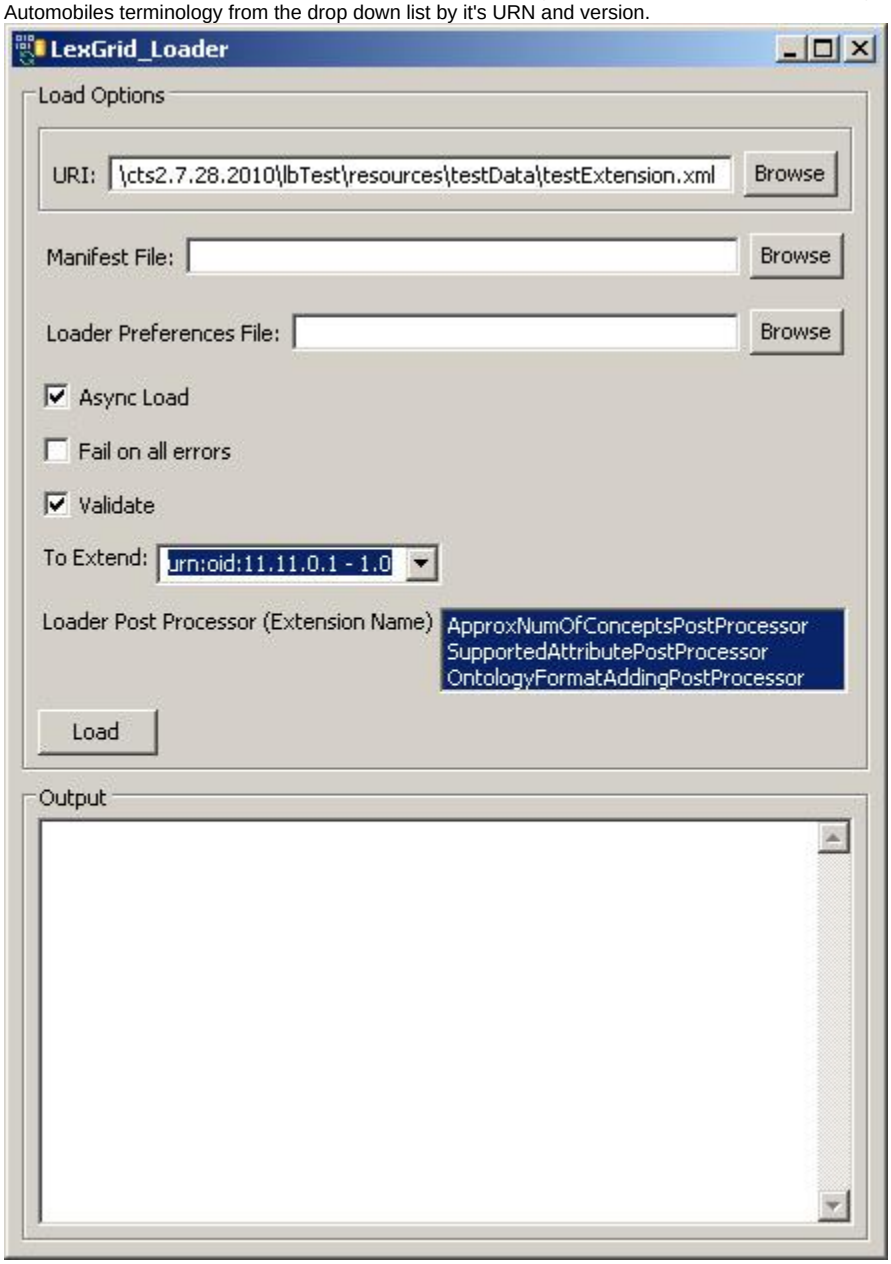
2. Load from <LexEVS root>/test/resources/testData/ the coding scheme Automobiles.xml (You should be able to do this using a source in any format supported for loading)



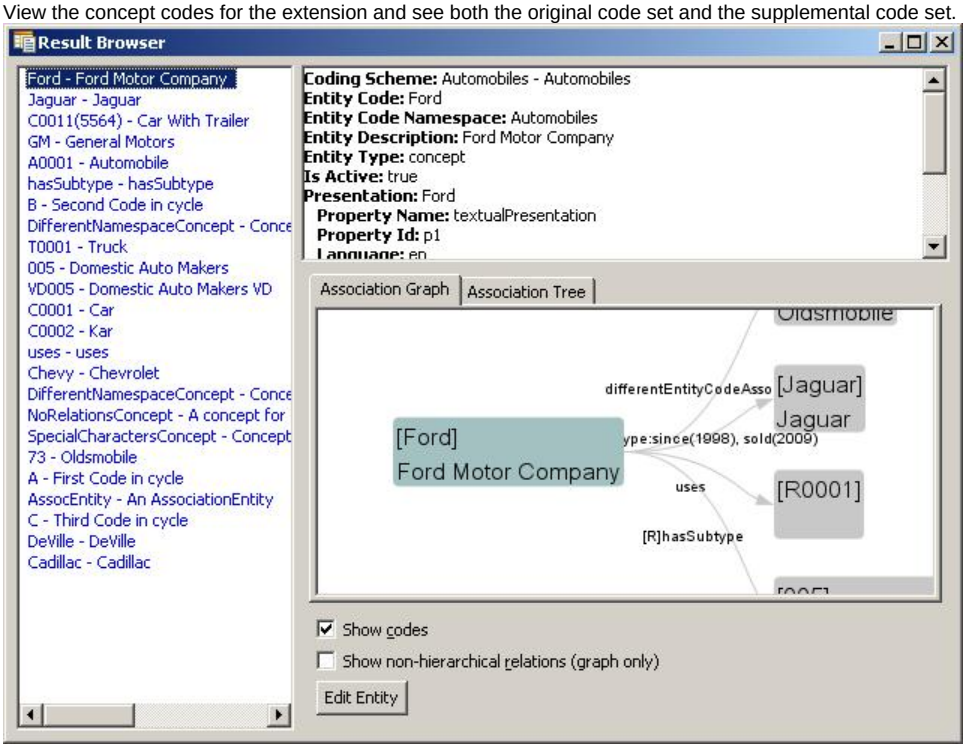
3. Activate this scheme and view it's contents by getting a coded node set and resolving it.



4. Load from <LexEVS root>/test/resources/testData/ the coding scheme testExtension.xml selecting the option to extend by selecting the Automobiles terminology from the drop down list by it's URN and version.



5. View the concept codes for the extension and see both the original code set and the supplemental code set.



Keep in mind that the testExtension.xml's file format can be used to extend any coding scheme currently loaded to LexEVS.

Authoring Coding Schemes and their Child Elements

A complete authoring API is featured in the CTS2 interface and is Detailed here

Create a Mapping Scheme using LexGrid XML

Mapping functions in LexEVS 6.: Inter-terminology relationship support, authoring and loading

What's Available in LexEVS 6.0:

- Automatic support for mapped values in sources that incorporate mapping
- Authoring capability for creating and updating mappings
- Loading support for mapping standard sources including MRMAP RRF files and LexGrid XML files.

Authoring Capability in LexEVS:

- Authoring API's in LexEVS 6.0
  - CTS2 Association Creation and Update
  - LexEVSAuthoringService for mappings and associations.
- Authoring a Mapping Coding Scheme in 6.0
  - Create a mapping scheme in LexGrid XML

Create a LexGrid Mapping Scheme:

- Requirements:
  - Address of the latest LexGrid model xsd: http://LexGrid.org/schema/2010/01/LexGrid/codingSchemes.xsd
  - An XML editor like oXygen.

Creating the Coding Scheme

- Open the XML editor and start a new document.
- Configure the editor to point to the desired root node: http://LexGrid.org/schema/2010/01/LexGrid/codingSchemes
- The editor may generate the root node, required attributes and any required children

Coding Scheme Elements

- We must populate these attributes:
  - CodingSchemeName

    - URI

Supported Elements (mappings)

- Under the <mappings> tag
    - Create references to the elements this mapping scheme will support
- Create a supported association
    - Create an appropriate name such as "mapped_to" or "approximately_mapped_to"
- Create Supported coding schemes
    - One for this scheme
    - One for the source scheme
    - One for the target scheme
    - Populate the local Id and the uri with correct values
- Create a supported container name for this coding scheme
    - should be distinctive if there is more than one mapping (and therefore more than one relations container) in the coding scheme
    - for this implementation it will just be "relations"
- Create supported namespaces
    - Again One for this scheme
    - One for the source scheme
    - One for the target scheme
    - Populate the local id and the equivalent coding scheme attributes.
    - Equivalent coding scheme points to the coding scheme local id this namespace must represent.

Create a Container for the Mappings

- Create a relations element
    - Populate the container name, the isMapping attribute, and source and target coding schemes.
    - Create (your XML editor should insist on it) an AssociationPredicate as a child element.
    - Populate the associationName attribute with the local Id of the supported association you created.

Create an Association Source

- This will be a child element of the Association Predicate.
- Populate the sourceEntityCodeNamespace with the supported namespace local Id of the source coding scheme.
- Populate the sourceEntityCode with the unique identifier (enityCode in LexGrid) of the concept in the source coding scheme.

Create an Association Target

- This will be a child element of the AssociationSource.
- Populate the targetEntityCodeNamespace with the supported namespace local Id of the target coding scheme.
- Populate the targetEntityCode with the unique identifier (enityCode in LexGrid) of the concept in the target coding scheme.