3 - LexEVS 6.x Management and Administration API

Contents of this Page

LexEVS Administration Links

- Admin Guide Main Page
  - Admin with LexEVS GUI
  - Admin with Command Line
  - Management and Admin API
  - Advanced Vocab Admin
- LexEVS 6.0 Main Page
- LexEVS Current Release

Retrieving Services

LexEVS provides a number of services which are retrievable through the LexBIGService interface and the LexEvsServiceLocator interface. We list here a small sample including the more commonly used interfaces.

LexBIGService

HistoryService

```
LexBIGService lbsv = LexBIGServiceImpl.defaultInstance();

try {

        HistoryService historSvc = lbsv.getHistoryService(codingScheme);<

} catch (LBException e) {

        e.printStackTrace();

}
```

LexBIGServiceManager

```
//LexBIG Service Manager can accept credentials validation to
//preserve service security.
LexBIGService lbsv = LexBIGServiceImpl.defaultInstance();

try {

        LexBIGServiceManager svcManager = lbsv.getServiceManager(credentials);

} catch (LBException e) {
        e.printStackTrace();
}
```

LexEVSServiceMetaData

```
// User safe service meta data query interface.
LexBIGService lbsv = LexBIGServiceImpl.defaultInstance();

try {

        LexBIGServiceMetadata svcMetaData = lbsv.getServiceMetadata();

} catch (LBException e) {
        e.printStackTrace();
}
```

LexEvsServiceLocator

SystemResourceService

```
// The system resource service is commonly wrapped in LexBIGServiceManager calls
// and as such it prevents unauthorized users from removing coding schemes from the service.
SystemResourceService systemResourceService = LexEvsServiceLocator.getInstance().getSystemResourceService();
       // must be marked as inactive or pending before delete.
       systemResourceService.removeCodingSchemeResourceFromSystem(
               codingSchemeVersionReference.getCodingSchemeURN(),
               codingSchemeVersionReference.getCodingSchemeVersion());
```

LexEvsIndexOperations

Some index operations can be accomplished using the interface returned from the LexEvsServiceLocator.

```
                boolean isSingleIndex =
                        LexEvsServiceLocator.getInstance().getSystemResourceService().getSystemVariables().
getIsSingleIndex();

                if(! isSingleIndex) {
                        throw new RuntimeException("Lucene Clean Up can only be executed in Single Index
Mode.");
                }
                try {
                        List<AbsoluteCodingSchemeVersionReference> expectedList =
                                new ArrayList<AbsoluteCodingSchemeVersionReference>();

                        for(RegistryEntry entry :
                                LexEvsServiceLocator.getInstance().getRegistry().getAllRegistryEntriesOfType
(ResourceType.CODING_SCHEME)) {

                                AbsoluteCodingSchemeVersionReference ref = new
AbsoluteCodingSchemeVersionReference();
                                ref.setCodingSchemeURN(entry.getResourceUri());
                                ref.setCodingSchemeVersion(entry.getResourceVersion());

                                expectedList.add(ref);
                        }

                        LexEvsServiceLocator.getInstance().getLexEvsIndexOperations().cleanUp(expectedList,
reindexMissing);
                } catch (Exception e) {
                        throw new RuntimeException(e);
                }
```

EntityIndexService

Others operations require using the IndexServiceManager to retrieve a more specific index service

```
            AbsoluteCodingSchemeVersionReference acsvr = new  AbsoluteCodingSchemeVersionReference();

            acsvr.setCodingSchemeURN(css.getCodingSchemeURI());
            acsvr.setCodingSchemeVersion(css.getRepresentsVersion());

            LexEvsServiceLocator.getInstance().
            getIndexServiceManager().
                getEntityIndexService().dropIndex(acsvr);
```

LexBIGService Administration Methods

We've shown how the LexBIGService interface can retrieve other services, but it also provides some user safe administrative functions. These functions provide users with information about the service and overlap general user API functions, since they give users information about how to use the service.

Getting Filter Extensions

```
        // LexEVS has no built-in filter extensions. This method call
        // will only return filters users have created themselves.
                LexBIGService lbsv = LexBIGServiceImpl.defaultInstance();
                Enumeration<? extends ExtensionDescription> extDesc = lbsv.getFilterExtensions().
enumerateExtensionDescription();
                while(extDesc.hasMoreElements()){
                        System.out.println(extDesc.nextElement().getName());
                }
```

Getting Sort Algorithms

```
                LexBIGService lbsv = LexBIGServiceImpl.defaultInstance();
                Enumeration<? extends SortDescription> sortDesc = lbsv.getSortAlgorithms(null).
enumerateSortDescription();
                while(sortDesc.hasMoreElements()){
                        System.out.println(sortDesc.nextElement().getName());
                }
```

Getting Match Algorithms

```
        LexBIGService lbsv = LexBIGServiceImpl.defaultInstance();
                Enumeration<? extends ModuleDescription> modDesc = lbsv.getMatchAlgorithms().
enumerateModuleDescription();
                while(modDesc.hasMoreElements()){
                        System.out.println(modDesc.nextElement().getName());
                }
```

Sample Output

```
 SpellingErrorTolerantSubStringSearch
 StemmedLuceneQuery
 literalContains
 startsWith
 nonLeadingWildcardLiteralSubString
 literal
 WeightedDoubleMetaphoneLuceneQuery
 literalSubString
 DoubleMetaphoneLuceneQuery
 RegExp
```

Getting Generic Extensions

```
                LexBIGService lbsv = LexBIGServiceImpl.defaultInstance();
                Enumeration<? extends ExtensionDescription> extDesc = lbsv.getGenericExtensions().
enumerateExtensionDescription();
                while(extDesc.hasMoreElements()){
                        System.out.println(extDesc.nextElement().getName());

                }
```

Sample Output

```
 SupplementExtension
 LexBIGServiceConvenienceMethods
 ApproxNumOfConceptsPostProcessor
 MappingExtension
 SupportedAttributePostProcessor
 OntologyFormatAddingPostProcessor
```

Getting Service Information on all Schemes

```
            LexBIGService lbsv = LexBIGServiceImpl.defaultInstance();
            try {
                    CodingSchemeRenderingList schemes = lbsv.getSupportedCodingSchemes();
        for (CodingSchemeRendering csr : schemes.getCodingSchemeRendering()) {
            // Separator ...
            System.out.println("============================");
            CodingSchemeSummary css = csr.getCodingSchemeSummary();
            CodingScheme cs = lbsv.resolveCodingScheme(css.getCodingSchemeURI(), Constructors
                    .createCodingSchemeVersionOrTagFromVersion(css.getRepresentsVersion()));
            System.out.println(ObjectToString.toString(cs, "", 80));}

            } catch (LBException e) {
                    // TODO Auto-generated catch block
                    e.printStackTrace();
            }
```

Sample Output Excerpt

```
CodingScheme
   Name: MDR12_1_TO_CST95
   Formal Name: MDR12_1_TO_CST95
   URI: urn:oid:CL413321.MDR.CST
   Approximate Number of Concepts: 0
   Default Language: null
   Represents Version: 200909
   Local Names:
       Array of 1 items:
           MDR12_1_TO_CST95
   Mappings:
       MappingsSupportedAssociations:
               Array of 1 items:
                   SupportedAssociation
                       Local ID: mapped_to
                       Content: mapped_to
                       AssociationEntity CodingScheme: null
                       AssociationEntity EntityCodeNamespace: null
                       AssociationEntity EntityCode: null
           SupportedCodingSchemes: ................................................
```

Getting Single Coding Scheme Information

```
        public void run(){
                String codingScheme = "amino-acid.owl";
                CodingSchemeVersionOrTag versionOrTag = new CodingSchemeVersionOrTag();
                versionOrTag.setVersion("1.2");
                LexBIGService lbsv = LexBIGServiceImpl.defaultInstance();
                try {
                        CodingScheme cs = lbsv.resolveCodingScheme(codingScheme, versionOrTag );
            System.out.println(ObjectToString.toString(cs, "", 80));

                } catch (LBException e) {
                        // TODO Auto-generated catch block
                        e.printStackTrace();
                }
```

Output same as preceding but for designated coding scheme only.

Getting a Coding Scheme's Copyright Content

```
                String codingScheme = "amino-acid.owl";
                CodingSchemeVersionOrTag versionOrTag = new CodingSchemeVersionOrTag();
                versionOrTag.setVersion("1.2");
                LexBIGService lbsv = LexBIGServiceImpl.defaultInstance();
                try {
                        String cpywrt = lbsv.resolveCodingSchemeCopyright(codingScheme, versionOrTag);
            System.out.println(cpywrt);

                } catch (LBException e) {
                        // TODO Auto-generated catch block
                        e.printStackTrace();
                }
```

Output is unformatted text

LexBIGService Manager Methods

These methods are generally not end user safe or access information in the system that is administrator specific in context. In general it should not be exposed to the end user.

Setting Coding Scheme Version Tag

Setting the Coding Scheme tag allows the system administrator to provide status tags or designate a preferred version of a given coding scheme for this terminology service.

```
AbsoluteCodingSchemeVersionReference codingSchemeVersion =
     new AbsoluteCodingSchemeVersionReference();
codingSchemeVersion.setCodingSchemeURN("http://www.co-ode.org/ontologies/amino-acid/2005/10/11
/amino-acid.owl");
codingSchemeVersion.setCodingSchemeVersion("1.0");

                LexBIGService lbsv = LexBIGServiceImpl.defaultInstance();

                try {
                        LexBIGServiceManager lbsvm = lbsv.getServiceManager(null);
                        // PRODUCTION tagged  vocabularies are interpreted by LexEVS
                        // to be the preferred version of the vocabulary.
                        lbsvm.setVersionTag(codingSchemeVersion, "PRODUCTION");

                } catch (LBException e) {
                        // TODO Auto-generated catch block
                        e.printStackTrace();
                }
```

Activate Coding Scheme

The active designation for a coding scheme is required to enable query functionality for the contained entities and their relationships and properties.

```
AbsoluteCodingSchemeVersionReference codingSchemeVersion =
    new AbsoluteCodingSchemeVersionReference();
codingSchemeVersion.setCodingSchemeURN("http://www.co-ode.org/ontologies/amino-acid/2005/10/1/amino-acid.owl");
codingSchemeVersion.setCodingSchemeVersion("1.0");
                LexBIGService lbsv = LexBIGServiceImpl.defaultInstance();

                try {
                        LexBIGServiceManager lbsvm = lbsv.getServiceManager(null);
                        lbsvm.activateCodingSchemeVersion(codingSchemeVersion);

                } catch (LBException e) {
                        // TODO Auto-generated catch block
                        e.printStackTrace();
```

Deactivate Coding Scheme

Allows administrator to restrict access to coding scheme metadata only.

```
AbsoluteCodingSchemeVersionReference codingSchemeVersion =
    new AbsoluteCodingSchemeVersionReference();
codingSchemeVersion.setCodingSchemeURN("http://www.co-ode.org/ontologies/amino-acid/2005/10/1/amino-acid.owl");
codingSchemeVersion.setCodingSchemeVersion("1.0");
                LexBIGService lbsv = LexBIGServiceImpl.defaultInstance();

                try {
                        LexBIGServiceManager lbsvm = lbsv.getServiceManager(null);
                        lbsvm.deactivateCodingSchemeVersion(codingSchemeVersion, null);

                } catch (LBException e) {
                        // TODO Auto-generated catch block
                        e.printStackTrace();
```

Remove Coding Scheme

Removes a coding scheme and its indexes from a terminology service.

> **Warning**
>
> This is a slow, resource consuming activity when the service is configured for single table mode.

```
AbsoluteCodingSchemeVersionReference codingSchemeVersion =
    new AbsoluteCodingSchemeVersionReference();
codingSchemeVersion.setCodingSchemeURN("http://www.co-ode.org/ontologies/amino-acid/2005/10/1
/amino-acid.owl");
codingSchemeVersion.setCodingSchemeVersion("1.0");
                LexBIGService lbsv = LexBIGServiceImpl.defaultInstance();

                try {
                        LexBIGServiceManager lbsvm = lbsv.getServiceManager(null);
                        lbsvm.removeCodingSchemeVersion(codingSchemeVersion);

                } catch (LBException e) {
                        // TODO Auto-generated catch block
                        e.printStackTrace();
                {
```

Remove History Service

Remove a history service from the terminology service.

```
                String codingScheme = "NCI Thesaurus";
                LexBIGService lbsv = LexBIGServiceImpl.defaultInstance();

                try {
                        LexBIGServiceManager lbsvm = lbsv.getServiceManager(null);
                        lbsvm.removeHistoryService(codingScheme);

                } catch (LBException e) {
                        // TODO Auto-generated catch block
                        e.printStackTrace();
                }
```

Get Load Extensions

Allows a determination of available loaders by an administrator.

```
LexBIGService lbsv = LexBIGServiceImpl.defaultInstance();

try {
        LexBIGServiceManager lbsvm = lbsv.getServiceManager(null);
         Enumeration<? extends ExtensionDescription> extDesc =
         lbsvm.getLoadExtensions().enumerateExtensionDescription();
         while(extDesc.hasMoreElements()){
                System.out.println(extDesc.nextElement().getName());
         }
} catch (LBException e) {
        // TODO Auto-generated catch block
        e.printStackTrace();
}
```

Sample Output

```
NCIMetaHistoryLoader
OBOLoader
HL7Loader
ClaMLLoader
SemNetLoader
MrMap_Loader
MetaDataLoader
RadLexFramesLoader
TextLoader
NCIThesaurusHistoryLoader
OWLLoader
LexGrid_Loader
```

Get Specific Loader

Returns a specific loader instance for use in loading the relevant source file.

```
LexBIGService lbs = LexBIGServiceImpl.defaultInstance();
LexBIGServiceManager lbsm = lbs.getServiceManager(null);
LexGrid_Loader loader = (LexGrid_Loader) lbsm
        .getLoader(org.LexGrid.LexBIG.Impl.loaders.LexGridMultiLoaderImpl.name);
    loader.setCodingSchemeManifestURI(manifest);
    loader.load(source, false, true);
```

Get Extension Registry

Returns an extension registry service for all externally registered plugins to LexEVS. This provides similar functionality to the other registry retrieval, display and use and we won't repeat method examples here.

```
LexBIGServiceManager lbsvm = lbsv.getServiceManager(null);
 ExtensionRegistry registry = lbsvm.getExtensionRegistry();
```

Get Export Extensions

Returns a list of Export extensions available as administrative functions to allow administrators to export a terminology to a specifically formatted output file.

```
LexBIGService lbsv = LexBIGServiceImpl.defaultInstance();

try {
        LexBIGServiceManager lbsvm = lbsv.getServiceManager(null);
        Enumeration<? extends ExtensionDescription> extDesc =
        lbsvm.getExportExtensions().enumerateExtensionDescription();
        while(extDesc.hasMoreElements()){
                System.out.println(extDesc.nextElement().getName());
        }

} catch (LBException e) {
        // TODO Auto-generated catch block
        e.printStackTrace();
}
```

Sample Output

```
LexGridExport
OwlRdfExporter
OBOExport
```

Get Exporter

Returns an exporter for a specific output format.

```
LexBIGService lbs = LexBIGServiceImpl.defaultInstance();
LexBIGServiceManager lbsm = lbs.getServiceManager(null);

// Find the registered extension handling this type of export ...
OwlRdfExporterImpl exporter = (OwlRdfExporterImpl) lbsm.getExporter(OwlRdfExporterImpl.name);

// Perform the requested action ...
CnsCngPair cngCngPair = FilterParser.parse(lbs, css.getCodingSchemeURI(),
css.getRepresentsVersion(), cl);
exporter.setCng(cngCngPair.getCng());
exporter.setCns(cngCngPair.getCns());
exporter.export(Constructors.createAbsoluteCodingSchemeVersionReference(css),
destination,overwrite,false, true);
```

Remove Metadata

Remove external coding scheme metadata.

```
    String urn = "urn:oid:11.00.11.1";
    String ver = "1.1";
    Enumeration<? extends CodingSchemeRendering> schemes = lbs.getSupportedCodingSchemes()
            .enumerateCodingSchemeRendering();
    while (schemes.hasMoreElements() && css == null) {
        CodingSchemeSummary summary = schemes.nextElement().getCodingSchemeSummary();
        if (urn.equalsIgnoreCase(summary.getCodingSchemeURI())
                && ver.equalsIgnoreCase(summary.getRepresentsVersion()))
            css = summary;
    }
}


    lbs.getServiceManager(null).removeCodingSchemeVersionMetaData(
            Constructors.createAbsoluteCodingSchemeVersionReference(css));
```

Register Coding Scheme Supplement

Register a coding scheme as a supplement to another coding scheme.

```
		try {
			LexBIGServiceManager lbsvm = lbsv.getServiceManager(null);
			AbsoluteCodingSchemeVersionReference parentCodingScheme  = new
			AbsoluteCodingSchemeVersionReference();
			AbsoluteCodingSchemeVersionReference supplementCodingScheme = new
			AbsoluteCodingSchemeVersionReference();
			String codingSchemeURN = "urn:oid:11.11.0.1.1";
			String codingSchemeVersion = "1.0";
			String ExtensionCodingSchemeURN = "urn:oid:11.11.0.1.1-extension";
			String ExtensionCodingSchemeVersion = "1.0-extension";
			parentCodingScheme.setCodingSchemeURN(codingSchemeURN);
			parentCodingScheme.setCodingSchemeVersion(codingSchemeVersion);
			supplementCodingScheme.setCodingSchemeURN(ExtensionCodingSchemeURN);
			supplementCodingScheme.setCodingSchemeVersion(ExtensionCodingSchemeVersion);

			lbsvm.registerCodingSchemeAsSupplement(parentCodingScheme,supplementCodingScheme);

		} catch (LBException e) {
			// TODO Auto-generated catch block
			e.printStackTrace();
		}
```

Remove a coding schemes relationship with another as a supplement.

Un-Register Coding Scheme Supplement

```
try {
	LexBIGServiceManager lbsvm = lbsv.getServiceManager(null);
	AbsoluteCodingSchemeVersionReference parentCodingScheme  = new
	AbsoluteCodingSchemeVersionReference();
	AbsoluteCodingSchemeVersionReference supplementCodingScheme = new
	AbsoluteCodingSchemeVersionReference();
	String codingSchemeURN = "urn:oid:11.11.0.1.1";
	String codingSchemeVersion = "1.0";
	String ExtensionCodingSchemeURN = "urn:oid:11.11.0.1.1-extension";
	String ExtensionCodingSchemeVersion = "1.0-extension";
	parentCodingScheme.setCodingSchemeURN(codingSchemeURN);
	parentCodingScheme.setCodingSchemeVersion(codingSchemeVersion);
	supplementCodingScheme.setCodingSchemeURN(ExtensionCodingSchemeURN);
	supplementCodingScheme.setCodingSchemeVersion(ExtensionCodingSchemeVersion);

			lbsvm.registerCodingSchemeAsSupplement(parentCodingScheme,supplementCodingScheme);

} catch (LBException e) {
	// TODO Auto-generated catch block
	e.printStackTrace();
}
```

LexEVS Authoring Service Management Methods

Currently restricted to the support of Mapping Authoring only.

Create a Mapping Coding Scheme from Scratch

```
     LexEVSAuthoringService authoring = new LexEVSAuthoringServiceImpl();

           EntryState entryState = new EntryState();
           entryState.setChangeType(ChangeType.NEW);
           entryState.setContainingRevision("FirstVersionNCItToICD9_12_1_54");
           entryState.setRelativeOrder(new Long(1));

                //Minimum meta data for the mapping coding scheme should be defined here
                AbsoluteCodingSchemeVersionReference mappingCodingScheme = new
AbsoluteCodingSchemeVersionReference();
                mappingCodingScheme.setCodingSchemeURN("urn:oid:11.0011.1.1");
                mappingCodingScheme.setCodingSchemeVersion("1.0");

                AbsoluteCodingSchemeVersionReference sourceCodingScheme = new
AbsoluteCodingSchemeVersionReference();
                sourceCodingScheme.setCodingSchemeURN("http://ncicb.nci.nih.gov/xml/owl/EVS/Thesaurus.owl#");
                sourceCodingScheme.setCodingSchemeVersion("10.10e");

                AbsoluteCodingSchemeVersionReference targetCodingScheme = new
AbsoluteCodingSchemeVersionReference();
                targetCodingScheme.setCodingSchemeURN("urn:oid:2.16.840.1.113883.6.2");
                targetCodingScheme.setCodingSchemeVersion("200909");

                AssociationSource[] associationSource = new AssociationSource[2];


                String associationType = "mapped_to";
                String relationsContainerName = "relations";
                Date effectiveDate = new Date();

                Revision revision = new Revision();
                revision.setRevisionId(entryState.getContainingRevision());

                boolean loadEntities = false;
                try {
                        authoring.createAssociationMapping(
                                        entryState,
                                    mappingCodingScheme,
                                        sourceCodingScheme,
                                        targetCodingScheme,
                                        associationSource,
                                        associationType,
                                        relationsContainerName,
                                        effectiveDate,
                                        null,
                                        revision,
                                        false
                                        );
                } catch (LBException e) {
                        // TODO Auto-generated catch block
                        e.printStackTrace();
                }
```

Create a Mapping Coding Scheme with Default Coding Scheme Values

```
                authoring.createMappingWithDefaultValues(sources, "NCI Thesaurus",
                                "10.10a", "ICD9", "200909", "mapped_to", false);
                AbsoluteCodingSchemeVersionReference codingSchemeVersion = new
                AbsoluteCodingSchemeVersionReference();
                codingSchemeVersion
                                .setCodingSchemeURN("http://default.mapping.container");
                codingSchemeVersion.setCodingSchemeVersion("1.0");
                lbsm.activateCodingSchemeVersion(codingSchemeVersion);
```

Insert Entity

```
private VersionableEventEntityService service;
 private AuthoringService authoringService;

 CodingScheme scheme = new CodingScheme();
 scheme.setApproxNumConcepts(111l);
 scheme.setCodingSchemeName("testName");
 scheme.setCodingSchemeURI("uri");
 scheme.setRepresentsVersion("v1");

 authoringService.loadRevision(scheme, null, null);

 CodingScheme cs = codingSchemeservice.getCodingSchemeByUriAndVersion("uri", "v1");
 System.out.println(cs);

 Entity entity = new Entity();
 entity.setEntityCode("c1");
 entity.setEntityCodeNamespace("ns");

 service.insertEntity("uri", "v1", entity);
```

Updating an Existing Entity

```
 private VersionableEventEntityService service;
private AuthoringService authoringService;


CodingScheme scheme = new CodingScheme();
scheme.setApproxNumConcepts(111l);
scheme.setCodingSchemeName("testName");
scheme.setCodingSchemeURI("uri");
scheme.setRepresentsVersion("v1");

authoringService.loadRevision(scheme, null, null);

CodingScheme cs = codingSchemeservice.getCodingSchemeByUriAndVersion("uri", "v1");
System.out.println(cs);

// Create a new entity
Entity entity = new Entity();
entity.setEntityCode("c1");
entity.setEntityCodeNamespace("ns");
entity.setIsDefined(false);

EntryState entryState = new EntryState();
entryState.setChangeType(ChangeType.MODIFY);
entity.setEntryState(entryState);

EntityDescription ed = new EntityDescription();
ed.setContent("pre-update");
entity.setEntityDescription(ed);

// Insert the new entity
service.insertEntity("uri", "v1", entity);

// Update the entity description
entity.getEntityDescription().setContent("post-update");

try {
 // Update the entity
 service.updateEntity("uri", "v1", entity);
} catch (Exception e) {

}

Entity modifiedEntity = service.getEntity("uri", "v1", "c1", "ns");
```

CTS2 Management Methods

No further information provided

Pick List and Value Set Managment Methods

No further information provided