LexEVS 6.x Migration Tips for 5.x Users

Contents of this Page

Introduction

LexEVS 6.0 functionality has been updated to provide many new features for terminology services. In the Tool Overview on the LexEVS 6.0 page, you can see an outline of this functionality. Changes were necessary to the existing LexEVS API and have been documented in the sections below. For help with the new APIs, see the LexEVS 6.0 Programmer's Guide. Before reading about each change, review the "What To Look For" section, which proposes a methodology that will help you find changes required in your code.

LexEVS 6.1 offers few changes to the Java API but an important update is the inclusion of a global search to the terminology service. The biggest change is the addition of several new loaders and a new REST service implementation that is completely CTS2 compliant.

Documentation

LexEVS 6.x REST CTS2 Service

Documentation of the CTS2 REST implementation in 6.1 and the legacy CTS2 Java API in 6.0. Included is the REST bulk download extension in 6.1.

LexEVS 6.x Installation Guide

This document outlines the supported configurations and technical installation instructions for LexEVS Vocabulary Services for caBIG®.

LexEVS 6.x Administration Guide

This guide outlines the environment configuration from the perspective of an existing installation.

LexEVS 6.x Programmer's Guide

This guide explains the LexEVS API (services, extensions, utilities, and GUI); also many related APIs.

LexEVS 6.x Design and Architecture Guide

This guide explains the LexGrid model and the LexBIG services.

LexEVS 6.x Value Set and Pick List Definition Guide

This guide explain the LexEVS 6.0 Value Set and Pick List Definition documentation.

LexEVS 6.x Loader Guide

This guide is intended for a LexEVS developer and provides information about the loaders provided, mapping, and how to create your own loaders using the loader framework.

Downloads

To access the download files please visit: LexEVS 6.0 API Downloads

What To Look For

In the following sections you will find documentation on what may need to be changed in your code when moving from LexEVS 5 to LexEVS 6. Before you go to each section, you can search your code for these strings (without the surrounding quotes). If you get a hit or hits, then you will need to make some changes in that area of the code. Follow the link for the string that you got a hit on in order to determine what to do in each case. This list may not catch everything, but it is a great place to start.

- Iterators changes according to the context of the Iterator.
- New Concept replaced by a "new Entity".
- After New Entity use the new addEntityType() method to set the specific Entity type.
- New Association replaced by new AssociationEntity and new AssociationPredicate.
- ValueDomain renamed to "ValueSet". This affects package names, class names, method names, method's input/output type/names.
- LexEVSValueDomainServices renamed to LexEVSValueSetServices.
- Method resolveValueDomain in LexEVSValueDomainServices, renamed to resolveValueSetDefinition and the method's arguments have changed.
- LexEVSPickListServices renamed to LexEVSPickListDefinitionServices
- Method resolvePickList in LexEVSPickListDefinitionServices, arguments have changed.

API Changes

Iterators

For example Iterator<ResolvedConceptReference>

```
Iterator<ResolvedConceptReference>
```

may need to be replaced by

```
Iterator<? extends ResolvedConceptReference>
```

Concept vs Entity

In 5.0 org.LexGrid.concepts.Concept and org.LexGrid.concepts.Instance were a subclass of org.LexGrid.concepts.Entity. These have been removed for 6.0. If you have used concept or instance in LexEVS 5.x ,then in 6.0 you will want to modify your code to use Entity instead.

In 5.0, for example, a concept can be defined as follows:

```
Concept concept = new Concept();
concept.setEntityCode("code");
concept.setEntityCodeNamespace("namespace");
concept.setIsDefined(true);
concept.setIsAnonymous(false);
concept.setIsActive(true);
```

In 6.0, a concept entity is defined as follows:

```
Entity concept = new Entity();
concept.addEntityType(EntityTypes.CONCEPT.toString());
concept.setEntityCode("code");
concept.setEntityCodeNamespace("namespace");
concept.setIsDefined(true);
concept.setIsAnonymous(false);
concept.setIsActive(true);
```

You may notice that we use the addEntityType method to specify the entity type of Concept. We use the same method for Instance. In 6.0 we define Instance as follows:

```
Entity instance = new Entity();
instance.addEntityType(EntityTypes.INSTANCE.toString());
instance.setEntityCode("code");
instance.setEntityCodeNamespace("namespace");
```

Association vs AssociationPredicate and AssociationEntity

org.LexGrid.relations.Association is replaced by AssociationPredicate and AssociationEntity in 6.0. The AssociationPredicate class contains 'associationName' and 'sourceList' properties, and their get/set methods. AssociationEntity class contains the properties such as 'forwardName', 'isNavigable', 'isTransitive', 'reverseName' etc., plus their get/set methods. For example, in 5.0 an Association class can be created as follows:

```
Association association = new Association();
association.setAssociationName("name");
association.addSource(source);
association.setForwardName("forwardName");
association.setIsNavigable(true);
```

In 6.0 it is replaced by

```
AssociationPredicate associationPredicate = new AssociationPredicate();
associationPredicate.setAssociationName("name");
associationPredicate.addSource(source);

AssociationEntity associationEntity = new AssociationEntity();
associationEntity.setEntityCode("name");
associationEntity.setForwardName("forwardName");
associationEntity.setIsNavigable(true);
```

Value Sets

- All class names that contain "ValueDomain" are now replaced with "ValueSet", for instance, ValueDomainDefinition is renamed to ValueSetDefinition.
- Major Value Set interfaces changes: These affect class names, method names, method's input/output type/names. If the name contains "ValueDomain", rename it to "ValueSet".
  - ValueDomainDefinition to ValueSetDefinition.
  - LexEVSValueDomainServices to LexEVSValueSetDefinitionServices.
- All packages that had "valuedomain" are now "valuesets".
- LexEVSPickListServices has been renamed to LexEVSPickListDefinitionServices
- Resolve supplied ValueSetDefinition object: In 5.0, this method is defined under LexEVSValueDomainServices.

  ```
  public ResolvedValueDomainDefinition resolveValueDomain(URI valueDomainURI,
         AbsoluteCodingSchemeVersionReferenceList csVersionList,
         String versionTag) throws LBException;
  ```

  In 6.0, the method is under LexEVSValueSetDefinitionServices. It is represented as follows:

  ```
  public ResolvedValueSetDefinition resolveValueSetDefinition(URI valueSetDefinitionURI,
         String valueSetDefinitionRevisionId,
         AbsoluteCodingSchemeVersionReferenceList csVersionList,
         String versionTag, SortOptionList sortOptionList) throws LBException;
  ```

  Besides the return type change, two new input arguments have been added. valueSetDefinitionRevisionId which holds the information of the version of the value set definition and sortOptionList which represents the list of sort options to apply during resolution. If supplied, the sort algorithms will be applied in the order provided. Any algorithms not valid to be applied in context of node set iteration, as specified in the sort extension description, will result in an argument exception. Available algorithms can be retrieved through the LexBIGService getSortExtensions method after being defined to the LexBIGServiceManager extension registry. For example:

```
AbsoluteCodingSchemeVersionReferenceList acsvList = null;
Util.displayMessage("Now select Code System to use to resolve Value Set Definition");
CodingSchemeSummary css = Util.promptForCodeSystem();
if (css != null)
{
 acsvList = new AbsoluteCodingSchemeVersionReferenceList();
 acsvList.addAbsoluteCodingSchemeVersionReference(Constructors.createAbsoluteCodingSchemeVersionReference
(css.getCodingSchemeURI(),
                                                  css.getRepresentsVersion()));
}

LexEVSValueSetDefinitionServices vsdServ = LexEVSValueSetDefinitionServicesImpl.defaultInstance();
ResolvedValueSetDefinition rVSD = null;
try {
 rVSD = vsdServ.resolveValueSetDefinition(new URI("myUri"), null, acsvList, null, null);
} catch (URISyntaxException e) {
 e.printStackTrace();
}
```

A new resolve value set definition method is provided. It accepts a ValueSetDefinition object instead of a URI.

```
public ResolvedValueSetDefinition resolveValueSetDefinition(ValueSetDefinition vsDef,
AbsoluteCodingSchemeVersionReferenceList csVersionList,
String versionTag, SortOptionList sortOptionList) throws LBException;
```

Pick Lists

- LexEVSPickListServices renamed to LexEVSPickListDefinitionServices
- Resolve supplied PickListDefinition object. In 5.0, LexEVSPickListServices provides this function. Two methods are defined as follows:

```
public ResolvedPickListEntryList resolvePickList(String pickListId, Integer sortType) throws LBException;
```

```
public ResolvedPickListEntryList resolvePickList(String pickListId, boolean sortByText) throws
LBException;
```

In 6.0, these methods are defined in LexEVSPickListDefinitionServices.java.

```
public ResolvedPickListEntryList resolvePickList(String pickListId, Integer sortType,
       AbsoluteCodingSchemeVersionReferenceList csVersionList, String versionTag) throws LBException;
```

```
public ResolvedPickListEntryList resolvePickList(String pickListId, boolean sortByText,
       AbsoluteCodingSchemeVersionReferenceList csVersionList, String versionTag) throws LBException;
```

Argument 'csVersionList' and 'versionTag' are added. 'csVersionList' represents a list of coding scheme URI's and versions to be used. These will be used only if they are present in the service. If absent, the most recent version will be used instead. 'versionTag' is the tag (e.g "devel", "production", ...) to be used to reconcile coding schemes when more than one is present. Note that non-tagged versions will be used if the tagged version is missing. An example using integer to indicate the sort type is shown as follows:

```
AbsoluteCodingSchemeVersionReferenceList incsvrl = new AbsoluteCodingSchemeVersionReferenceList();
incsvrl.addAbsoluteCodingSchemeVersionReference(Constructors.createAbsoluteCodingSchemeVersionReference
("urn:oid:11.11.0.1", "1.1"));

ResolvedPickListEntryList pls = _pickListService.resolvePickList("SRITEST:FA:
MicrobialStructureOntologyMinusMCell", 1, incsvrl, "production");
```

- Besides the two methods above, LexEVSPickListServices also adds two new methods resolvePickList and resolvePickListForTerm as follows:
- The new resolvePickList method can be passed in a PickListDefinition object instead of pickListId.

```
public ResolvedPickListEntryList resolvePickList(PickListDefinition pickList, boolean sortByText,
        AbsoluteCodingSchemeVersionReferenceList csVersionList,
        String versionTag) throws LBException;
```

```
public ResolvedPickListEntryList resolvePickListForTerm(String pickListId, String term, String
matchAlgorithm,
        String language, String[] context, boolean sortByText,A
        bsoluteCodingSchemeVersionReferenceList csVersionList, String versionTag) throws LBException;
```

- More restriction arguments have been added to the new resolvePickListForTerm method, such as: 'term', 'matchAlgorithm', 'languange', and 'context'. The argument 'term' is required. An example is shown below:

```
ResolvedPickListEntryList pickLists = getPickListService().resolvePickListForTerm("SRITEST:AUTO:
AllDomesticButGM",
       "Jaguar", MatchAlgorithms.exactMatch.name(),
       "english", null, false, null, null);
```

- In 6.0 you also can resolve a pick list by revision. The code below is the new method:

```
public PickListDefinition resolvePickListByRevision(String pickListId,
       String revisionId, Integer sortOrder) throws LBRevisionException;
```

Note: the return type of this method is PickListDefinition instead of ResolvedPickListEntryList.

Configuration Changes

New lbconfig.props

- There are new parameters in the lbconfig.props file. Please see the LexEVS 6.x Local Runtime Configuration File Settings.
- Single or Multi-table set mode is now available in LexEVS 6.0. This allows users to either load multiple coding schemes into one set of tables (single table set mode) or each coding scheme has its own set of tables (multi table set mode). Please see the LexEVS 6.x Local Runtime Configuration File Settings to ensure your settings are correct for your installation. If you do not make a selection single table set mode will be used.

New or Updated Jars

- ddlUtils-1.0-PATCHED.jar
- ibatis-sqlmap-2.3.4.726.jar
- hibernate-annotations-3.4.0.GA.jar
- hibernate-commons-annotations-3.3.0.ga.jar
- persistence-api-1.0.jar
- hibernate-core-3.3.2.GA.jar
- ehcache-core-2.0.1.jar
- commons-betwixt-0.8.jar
- commons-digester-1.8.jar
- compass-2.2.0.jar
- lucene-analyzers.2.4.1.jar
- lucene-queries-2.4.1.jar
- castor-1.3.1-anttasks.jar
- castor-1.3.1-codegen.jar
- castor-1.3.1-core.jar
- castor-1.3.1-xml-schema.jar
- castor-1.3.1.jar

Model Changes

- ValueDomains changes to ValueSets
- ValueDomainDefinition changed to ValueSetDefinition
- Added ConceptDomain attribute to ValueSetDefinition
- Renamed PickListDefinition's attribute completeDomain to CompleteSet