LexEVS 6.x Pick List Service

Contents of this Page

LexEVS Value Set Links

- Value Set Guide Main Page
  - Value Set Design
  - Value Set Service API
  - Pick List Design
  - Pick List Service API
  - Value Set GUI
- Programmer's Guide Main Page
- LexEVS 6.0 Main Page
- LexEVS Current Release

Introduction

The Pick List services are integrated parts of the LexEVS API. It provide three major functions:

- **Administration**- Ability to load, export and remove Pick List Definitions
  - Loader - Ability to load Pick List Definitions programmatically into the LexGrid repository using the domain objects that are available via the LexGrid logical model
  - Exporter - Ability to export Pick List Definition and Pick List Resolution to a file in LexGrid XML format
  - Remove - Ability to remove Pick List Definition from the system

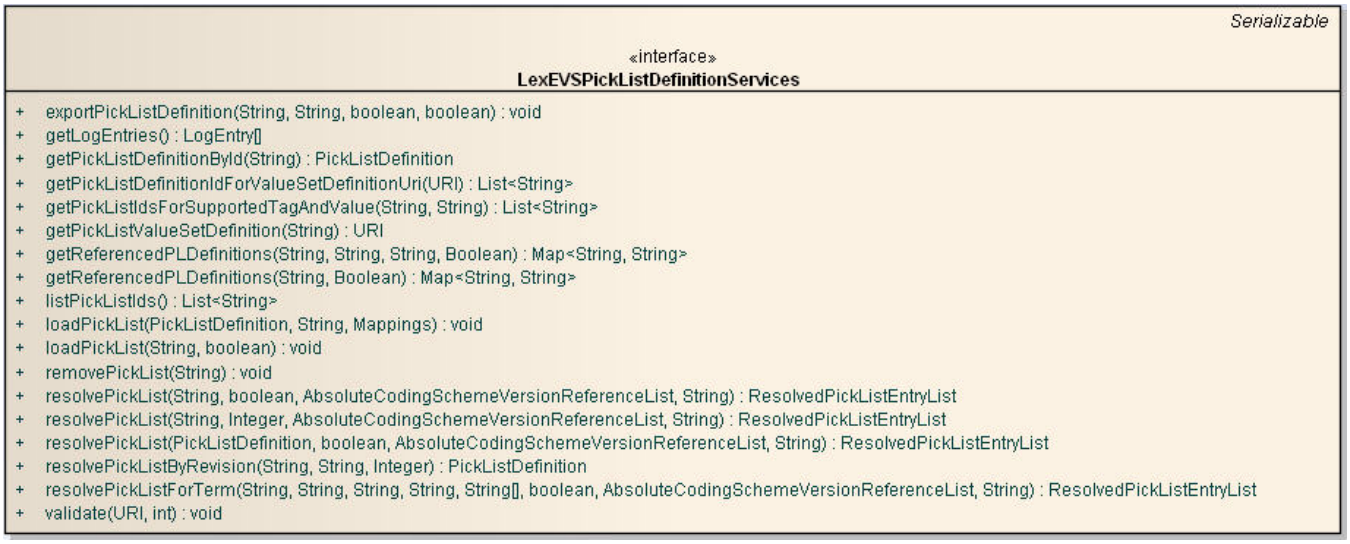
> **Note**
>
> These administration operations can be performed in LexEVS 'Local' environment only. This can not be performed using LexEVS Distributed environment.

- **Query** - Ability to apply user restrictions (ex: PickListId) and dynamically resolve the definitions at run time
- **Resolve** - Ability to resolve Pick List Definition dynamically against selected Coding Scheme Version(s) and return all/selected terms of the concepts belonging to the pick list

The LexEVS Pick List Services expose the API particularly for the Pick List Definition elements of the LexGrid Logical Model. For more information on LexGrid model see the LexGrid model schema

LexEVSPickListDefinitionServices Class Diagram

LexEVSPickListDefinitionServices is the main interface for all the services provided by LexEVS Pick List API. Here is the class diagram of LexEVSPickListDefinitionServices:



LexEVS Pick List Service API

LexEVS Pick List Definition Services provides three major functions:

- Administration function
- Query function and
- Resolve function

Each of these functions are described in following sections.

Administration Functions

LexEVS Pick List Definition Services provides following administration functions :

- Load
- Export and
- Remove

> **Note**
>
> These administration operations can be performed in LexEVS 'Local' environment only. This can not be performed using LexEVS Distributed environment.

Load Functions

- Loading Pick List Definition Object - This function provides the capability to load supplied PickListDefinition object into the system.
- Loading Pick List Definitions in file - This function provides the capability to load Pick List Definitions found in file that are in LexGrid XML format.

Loading Pick List Definition Object

This function can be used to load supplied Pick List Definition object to the system.

```
loadPickList(PickListDefinition pldef, String systemReleaseURI, Mappings mappings)
```

Description:	Loads supplied Pick List Definition object
Input:	*org.LexGrid.valueSets.PickListDefinition* pldef - (Mandatory) pick list definition object to load *java.lang.String* systemReleaseURI - (Optional) System Release URI this Pick List Definition belongs to. *org.LexGrid.naming.Mappings* mappings - (Optional) List of Supported Attributes this Pick List Definition uses.
Output:	none
Exception:	*LBException*

Implementation Details:	***Implementation:***
	Step 1: Call this method on the associated LexEVS Pick List Service instance to load the Pick List Definition object and the System Release URI that this definition belongs to.
	Sample Call:
	• *Step 1:*Instantiate LexEVSPickListServices if it is not done yet :
	`org.lexgrid.valuesets.LexEVSPickListDefinitionServices plServ = org.lexgrid.valuesets.impl.LexEVSPickListDefinitionServicesImpl.defaultInstance();`
	• *Step 2:*Create and populate the PickListDefinition object.
	`org.LexGrid.valueSets.PickListDefinition pickList = new org.LexGrid.valueSets.PickListDefinition();`
	`pickList.setPickListId(pickListId);`
	`pickList.setRepresentsValueSetDefinition(vdURI);`
	`pickList.setCompleteSet(true);`
	`pickList.setDefaultEntityCodeNamespace(ecns);`
	`pickList.setDefaultLanguage("en");`
	`pickList.setDefaultSortOrder("asc");`
	`pickList.setIsActive(true);`
	`pickList.setEntityDescription(red);`
	Similarly, PickListEntryNode, Property, and Mapping objects can be created and assigned to the pickList object.
	`_pickList.getPickListEntryNode.add(pickListEntry);`
	`_pickList.setProperties(propertisObject);`
	`_pickList.setMappings(mappingsObject);`
	• *Step 3:*Call the load method by passing the Pick List Definition object and the System Release URI.
	`plServ.loadPickList(pickList,"Release 2010", null);`

Loading Pick List Definitions in File

This function can be used to load Pick List Definition(s) to the system from a file that is in LexGrid XML format.

`loadPickList(String xmlFileLocation, boolean failOnAllErrors)`

Description:	Loads Pick List Definitions found in input xml file.
Input:	*java.lang.String* xmlFileLocation - (Mandatory) File containing Pick List Definition(s) in LexGrid XML format. *boolean* failOnErrors - (Optional) default is false.
Output:	none
Exception:	*LBException*

Implementation Details:	***Implementation:***
	Step 1: Call this method on the associated LexEVS Pick List Service instance to load all Pick List Definitions found in an XML file that is in LexGrid format.
	Sample Call:
	• *Step 1:*Instantiate LexEVSPickListServices if it is not done yet :
	`org.lexgrid.valuesets.LexEVSPickListDefinitionServices plServ = org.lexgrid.valuesets.impl.LexEVSPickListDefinitionServicesImpl.defaultInstance();`
	• *Step 2:*Call load method by passing the inputfile location and boolean flag whether to stop on load errors:
	`plServ.loadPickList(inputXMLFile, true);`

Export Pick List

The only export function available :

- Export Pick List Definition - This function provides the capability to export Pick list Definition to a file in LexGrid XML format. This will be helpful if there is a need to import this exported Pick List Definition in different instance of LexEVS.

Export Pick List Definition

This function provides the capability to export Pick List Definition to a file in LexGrid XML format. This will be helpful if there is a need to import this exported Pick List Definition in different instance of LexEVS.

`exportPickListDefinition(String pickListId, String xmlFolderLocation, boolean overwrite, boolean failOnAllErrors)`

Description:	Export Pick List Definition to a file in LexGrid XML format.

Input:	*java.lang.String* pickListId- (Optional) id of pick list definition to export
	java.lang.String xmlFullPathName - (Mandatory) Location to save the definition
	`boolean  overwrite - True: to override the existing file.`
	`boolean failOnAllErrors - True: stops exporting if any error. |`

Output:	none
Exception:	*org.LexGrid.LexBIG.Exceptions.LBException*

Implementation Details:	***Implementation:***
	_Step 1:_Call this method on the associated LexEVS Pick List Service instance to export the Pick List Definition to a file in LexGrid XML format.
	Sample Call:
	• *Step 1:*Instantiate LexEVSPickListDefinitionServices if it is not done yet :
	`org.lexgrid.valuesets.LexEVSPickListDefinitionServices plServ = org.lexgrid.valuesets.impl.LexEVSPickListDefinitionServicesImpl.defaultInstance();`
	• *Step 2:*Call exportPickListDefinition method and provide parameter values :
	`plServ.exportPickListDefinition(valueSetDefinitionURI, null, true, false);`

Remove Pick List Definition

This function provides the capability to remove pick list definition from the system.

`removePickList(String pickListId)`

Description:	Removes supplied Pick List Definition from the system.
Input:	*java.lang.String*
Output:	none
Exception:	*org.LexGrid.LexBIG.Exceptions.LBException*

Implementation Details:	***Implementation:***
	Step 1: Call this method on the associated LexEVS Pick List Service instance to remove Pick List Definition from the system that matches the supplied pickListId.
	Sample Call:
	• *Step 1:*Instantiate LexEVSPickListServices if it is not done yet :
	`org.lexgrid.valuesets.LexEVSPickListServices plServ = org.lexgrid.valuesets.impl.LexEVSPickListDefinitionServicesImpl.defaultInstance();`
	• *Step 2:*Call removePickList method:
	`plServ.removePickList ("AUTO:AllDomesticANDGM");`

Query Functions

Here are some of the query functions that can be run against pick list definitions using the LexEVS Pick List Services.

Validate XML Resources

`validate(URI uri, int valicationLevel)`

Description:	Perform validation of the candidate resource without loading data.
Input:	*java.net URI* xmlFileLocation - (Mandatory) File containing PickListDefinition(s) in LexGrid XML format. *int* validationLevel_ - validatation level includes: • 0 = Verify document is well-formed • 1 = Verify document is valid
Output:	none
Exception:	*Org.LexGrid.LexBIG.Exceptions.LBParameterException*

Implementation Details:	***Implementation:***
	Step 1: Call this method on the associated LexEVS Pick List Service instance to validate the XML file that is in LexGrid format. This call will not load the data in an XML file.
	Sample Call:
	• *Step 1:*Instantiate LexEVSPickListServices if it is not done yet :
	`org.lexgrid.valuesets.LexEVSPickListServices plServ = org.lexgrid.valuesets.impl.LexEVSPickListDefinitionServicesImpl.defaultInstance();`
	• *Step 2:*Call validate method for validation by supplying URI of the XML file and validation level.
	`plServ.validate(uriOfXMLFile, true);`

getPickListDefinitionById

`getPickListDefinitionById(String pickListId)`

Description:	Returns pickList definition for supplied pickListId.
Input:	*java.lang.String* pickListId - (Mandatory) Id of a pickListDefinition
Output:	*org.LexGrid.valueSets.PickListDefinition* - Pick List Definition Object
Exception:	*org.LexGrid.LexBIG.Exceptions.LBException*

Implementation Details:	***Implementation:***
	Step 1: Call this method on the associated LexEVS Pick List Service instance to get Pick List Definition for supplied pickListId.
	Sample Call:
	• *Step 1:*Instantiate LexEVSPickListServices if it is not done yet :
	`org.lexgrid.valuesets.LexEVSPickListServices plServ = org.lexgrid.valuesets.impl.LexEVSPickListDefinitionServicesImpl.defaultInstance();`
	• *Step 2:*Call getPickListDefinitionById method:
	`org.LexGrid.valueSets.PickListDefinition plDef = plServ.getPickListDefinitionById("AUTO:DomesticAutoMakers");`

getPickListDefinitionIdForValueSetDefinitionUri

`getPickListDefinitionIdForValueSetDefinitionUri(URI valueSetDefinitionURI)`

Description:	Returns all the pickList definition id's that represent the supplied valueSetDefinition URI.
Input:	*java.net.URI* valueSetDefinitionURI - (Mandatory) URI of an value set definition
Output:	*List<String>* - List of Pick List Definition Id's that represents supplied valueSetDefURI.
Exception:	*org.LexGrid.LexBIG.Exceptions.LBException*

Implementation Details:	***Implementation:***
	Step 1: Call this method on the associated LexEVS Pick List Service instance to get all the Pick List Definitions that are represented by supplied Value Set Definition URI.
	Sample Call:
	• *Step 1:*Instantiate LexEVSPickListServices if it is not done yet :
	`org.lexgrid.valuesets.LexEVSPickListServices plServ = org.lexgrid.valuesets.impl.LexEVSPickListDefinitionServicesImpl.defaultInstance();`
	• *Step 2:*Call getPickListDefinitionIdForValueSetDefinitionUri method:
	`java.util.List<String> plDefs = pls.getPickListDefinitionIdForValueSetDefinitionUri(new URI("http://someDomain.html/ValueSets/NCIT/C3367_Value_set"));`

getPickListValueSetDefinition

`getPickListValueSetDefinition(String pickListId)`

Description:	Returns a URI of the represented value set definition of the pickList.
Input:	*java.lang.String* pickListId - (Mandatory) id of pick list definition
Output:	*java.net.URI* - Value Set Definition URI
Exception:	*org.LexGrid.LexBIG.Exceptions.LBException*

Implementation Details:	***Implementation:***
	Step 1: Call this method on the associated LexEVS Pick List Service instance to get a Value Set Definition URI represented by supplied pickListId.
	Sample Call:
	• *Step 1:*Instantiate LexEVSPickListServices if it is not done yet :
	`org.lexgrid.valuesets.LexEVSPickListServices plServ = org.lexgrid.valuesets.impl.LexEVSPickListDefinitionServicesImpl.defaultInstance();`
	• *Step 2:*Call getPickListValueSetDefinition method:
	`URI vsdURI = pls.getPickListValueSetDefinition("AUTO:DomesticAutoMakers");`

listPickListIds

`listPickListIds()`

Description:	Returns a list of pickListIds that are available in the system.
Input:	none
Output:	*java.util.List<java.lang.String>* - list of available pickListIds
Exception:	*org.LexGrid.LexBIG.Exceptions.LBException*

Implementation Details:	***Implementation:***
	Step 1: Call this method on the associated LexEVS Pick List Service instance to get all the PickListIds that are loaded in the system.
	Sample Call:
	• *Step 1:*Instantiate LexEVSPickListServices if it is not done yet :
	`org.lexgrid.valuesets.LexEVSPickListServices plServ = org.lexgrid.valuesets.impl.LexEVSPickListDefinitionServicesImpl.defaultInstance();`
	• *Step 2:*Call listPickListIds method:
	`java.util.List<String> plList = plServ.listPickListIds();`

Resolve Functions

Resolve functions provides the capability to resolve the pick list definition against specific coding scheme version and get back the list of terms from the entities that belongs to the pick list. If no coding scheme version is provided, the API will pick the latest version of the coding scheme.

There are two separate functions available to resolve pick list definition :

- Resolve Stored Pick List Definition - This function allows you to resolve pick list definition that are loaded in the system.
- Resolve Supplied Pick List Definition - This function allows you to pass a PickListDefinition object that may not be loaded in the system and get it resolved.

Resolving Stored Pick List Definition

This function resolves the Pick List Definition that is loaded in the system and returns set of valid terms of the concepts plus the code systems and its versions that were used.

resolvePickList

`resolvePickList(String pickListId, boolean sortByText, AbsoluteCodingSchemeVersionReferenceList csVersionList, String versionTag)`

Description:	Resolves pickList definition for supplied pickListId.

Input:	*java.langString* pickListId - (Mandatory) pickListId of a pickListDefinition. *boolean* sortByText - If True; the resolved pickListEntries will be sorted by text in ascending order. *org.LexGrid.LexBIG.DataModel.Collections.AbsoluteCodingSchemeVersionReferenceList* csVersionList - (Optional) 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. *java.lang.String* versionTag - (Optional) 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.
Output:	*org.lexgrid.valuesets.dto.ResolvedPickListEntryList* - Resolved PickListEntries
Exception:	*org.LexGrid.LexBIG.Exceptions.LBException*
Implementation Details:	***Implementation:*** *Step 1:* Call this method on the associated LexEVS Pick List Service instance to get the resolved Pick List Entries for the supplied pickListId. Optionally, if sortByTests is true, sort the pickText in the list. ***Sample Call:***

- *Step 1:*Instantiate LexEVSPickListServices if it is not done yet :

```
 org.lexgrid.valuesets.LexEVSPickListServices plServ = org.lexgrid.valuesets.impl.
LexEVSPickListDefinitionServicesImpl.defaultInstance();
```

- *Step 2:*Populate coding scheme versions list object:

```
 String codingSchemeName = "Automobiles";
 String version = "1.1";
 AbsoluteCodingSchemeVersionReferenceList csVersionList = new
AbsoluteCodingSchemeVersionReferenceList();
 csVersionList.addAbsoluteCodingSchemeVersionReference(Constructors.
createAbsoluteCodingSchemeVersionReference(codingSchemeName, version));
```

- *Step 3:*Call resolvePickList method:

```
 ResolvedPickListEntryList pleList = plServ.resolvePickList("AUTO:DomesticAutoMakers", true,
csVersionList, "PRODUCTION");
```

resolvePickList

`resolvePickList(String pickListId, Integer sortType, AbsoluteCodingSchemeVersionReferenceList csVersionList, String versionTag)`

Description:	This method is similar to above method which resolves pickList definition for supplied pickListId. In addition to sorting the terms in Ascending or Descending, this method can be used to custom sort the terms.
Input:	*java.langString* pickListId - (Mandatory) pickListId of a pickListDefinition. *java.lang.Integer* sortType - Sort type for pick text : 1-Ascending, 2-Descending, and 3-Custom *org.LexGrid.LexBIG.DataModel.Collections.AbsoluteCodingSchemeVersionReferenceList* csVersionList - (Optional) 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. *java.lang.String* versionTag - (Optional) 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.
Output:	*org.lexgrid.valuesets.dto.ResolvedPickListEntryList* - Resolved PickListEntries
Exception:	*org.LexGrid.LexBIG.Exceptions.LBException*

Implementation Details:	***Implementation:***
	Step 1: Call this method on the associated LexEVS Pick List Service instance to get the resolved Pick List Entries for the supplied pickListId. Optionally, pickText can be sorted in Ascending, Descending or in Custom order.
	Sample Call:
	• *Step 1:*Instantiate LexEVSPickListServices if it is not done yet :
	`org.lexgrid.valuesets.LexEVSPickListServices plServ = org.lexgrid.valuesets.impl.LexEVSPickListDefinitionServicesImpl.defaultInstance();`
	• *Step 2:*Populate coding scheme versions list object:
	`String codingSchemeName = "Automobiles";`
	`String version = "1.1";`
	`AbsoluteCodingSchemeVersionReferenceList csVersionList = new AbsoluteCodingSchemeVersionReferenceList();`
	`csVersionList.addAbsoluteCodingSchemeVersionReference(Constructors.createAbsoluteCodingSchemeVersionReference(codingSchemeName, version));`
	• *Step 3:*Call resolvePickList method:
	`ResolvedPickListEntryList pleList = plServ.resolvePickList("AUTO:DomesticAutoMakers", 3, csVersionList, "PRODUCTION");`

resolvePickListForTerm

`resolvePickListForTerm(String pickListId, String term, String matchAlgorithm, String language, String[] context, boolean sortByText, AbsoluteCodingSchemeVersionReferenceList csVersionList, String versionTag))`

Description:	Resolves pickList definition by applying supplied arguments.
Input:	*java.lang.String* pickListId - (Mandatory) pickListId of a pickListDefinition. This is required argument. *java.lang.String* term - (Mandatory) Term to restrict. This is required argument. *java.lang.String* matchAlgorithm - (Optional) match algorithm to use. Default - LuceneQuery *java.lang.String* language - (Optional) language to restrict. *java.lang.String[]* context - (Optional) list of context to restrict. *boolean* sortByText - If True; the resolved pickListEntries will be sorted by text in ascending order. *org.LexGrid.LexBIG.DataModel.Collections.AbsoluteCodingSchemeVersionReferenceList* csVersionList - (Optional) 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. *java.lang.String* versionTag - (Optional) 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.
Output:	*org.lexgrid.valuesets.dto.ResolvedPickListEntryList* - Resolved PickListEntries.
Exception:	*org.LexGrid.LexBIG.Exceptions.LBException*

Implementation Details:	***Implementation:***
	Step 1: Call this method on the associated LexEVS Pick List Service instance to get list of Pick List Entries that match the term supplied and meet other supplied restrictions.
	Sample Call:
	• *Step 1:*Instantiate LexEVSPickListServices if it is not done yet :
	`org.lexgrid.valuesets.LexEVSPickListServices plServ = org.lexgrid.valuesets.impl.LexEVSPickListDefinitionServicesImpl.defaultInstance();`
	• *Step 2:*Populate coding scheme versions list object:
	`String codingSchemeName = "Automobiles";`
	`String version = "1.1";`
	`AbsoluteCodingSchemeVersionReferenceList csVersionList = new AbsoluteCodingSchemeVersionReferenceList();`
	`csVersionList.addAbsoluteCodingSchemeVersionReference(Constructors.createAbsoluteCodingSchemeVersionReference(codingSchemeName, version));`
	• *Step 3:*Call resolvePickListForTerm method:
	`ResolvedPickListEntryList pleList = plServ.resolvePickListForTerm ("AUTO:DomesticAutoMakers","Jaguar", MatchAlgorithms.exactMatch.name(), "en", null, true, csVersionList, "PRODUCTION");`

Resolving Supplied Pick List Definition Object

This function resolves the supplied Pick List Definition object and returns set of valid terms of the concepts plus the code systems and its versions that were used.

resolvePickList

`resolvePickList(PickListDefinition pickList, boolean sortByText, AbsoluteCodingSchemeVersionReferenceList csVersionList, String versionTag)`

Description:	Resolves supplied pick list definition object.
Input:	*org.LexGrid.valueSets.PickListDefinition* pickList - (Mandatory) pickListDefinition object to resolve. *boolean* sortByText - If True; the resolved pickListEntries will be sorted by text in ascending order. *org.LexGrid.LexBIG.DataModel.Collections.AbsoluteCodingSchemeVersionReferenceList* csVersionList - (Optional) 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. *java.lang.String* versionTag - (Optional) 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.
Output:	*org.lexgrid.valuesets.dto.ResolvedPickListEntryList* - Resolved PickListEntries
Exception:	*org.LexGrid.LexBIG.Exceptions.LBException*

Implementation Details:	***Implementation:***
	Step 1: Call this method on the associated LexEVS Pick List Service instance to get the resolved Pick List Entries for the supplied pickListId. Optionally, if sortByTests is true, sort the pickText in the list.
	Sample Call:
	• *Step 1:* Instantiate LexEVSPickListServices if it is not done yet : *org.lexgrid.valuesets.LexEVSPickListServices plServ = org.lexgrid.valuesets.impl.LexEVSPickListDefinitionServicesImpl. defaultInstance();*
	• *Step 2:*Populate PickListDefinition object :
	`PickListDefinition pickList = new PickListDefinition();`
	`pickList.setCompleteSet(Boolean.TRUE);`
	`pickList.setDefaultEntityCodeNamespace("Automobiles");`
	`pickList.setDefaultLanguage("en");`
	`pickList.setRepresentsValueSetDefinition("SRITEST:AUTO:GM");`
	`pickList.setPickListId("PLObject");`
	`pickList.setStatus("active");`
	• *Step 2:*Populate coding scheme versions list object:
	`String codingSchemeName = "Automobiles";`
	`String version = "1.1";`
	`AbsoluteCodingSchemeVersionReferenceList csVersionList = new AbsoluteCodingSchemeVersionReferenceList();`
	`csVersionList.addAbsoluteCodingSchemeVersionReference(Constructors.createAbsoluteCodingSchemeVersionReference(codingSchemeName, version));`
	• *Step 3:*Call resolvePickList method:
	`ResolvedPickListEntryList pleList = plServ.resolvePickList(pickList, true, csVersionList, "PRODUCTION");`

Resolved Pick List Objects

Here are the resolved Objects from LexEVSPickListDefinitionServices :

- ResolvedPickListEntry : contains resolved Pick List Entry Nodes.
- ResolvedPickListEntryList : contains the list of resolved Pick List Entries. Also provides helpful features to add, remove, enumerate Pick List Entries.

Error Handling

LexEVS Pick List services uses org.LexGrid.LexBIG.Impl.loaders.MessageDirector to direct all fatal, error, warning, info messages with appropriate messages to the LexBIG log files in the 'log' folder of LexEVS install directory.

Along with MessageDirector, the services will also make use of org.LexGrid.LexBIG.exception.LBException to throw any fatal and error messages to the log file as well as to console.

Load Scripts

Scripts to load Pick List Definitions into LexEVS system will be located under 'Admin' folder of LexEVS install directory. These loader scripts will only load data in an XML file that is in LexGrid format.

LoadPickListDefinition.bat for Windows environment and
LoadPickListDefinition.sh for Unix environment.
Both of these scripts take in the following parameters:

```
 -in
 Input <uri>
 URI or path specifying location of the source file.

 -v
 Validate <int>
 Perform validation of the candidate resource without loading data. Supported levels of validation include:

0 = Verify document is well-formed
1 = Verify document is valid
```

Example:

```
 sh LoadPickListDefinition.sh \-in "file:///path/to/file.xml"
```

Sample Pick List Definitions XML File

The following is a sample XML file containing Pick List Definitions in LexGrid format that can be loaded using LexEVS Pick List Service.

```
<source> <?xml version="1.0" encoding="UTF-8"?>

<pickListDefinition pickListId="SRITEST:AUTO:DomesticAutoMakers"
representsValueSetDefinition="SRITEST:AUTO:DomesticAutoMakers"
isActive="true" defaultEntityCodeNamespace="Automobiles"
defaultLanguage="en" completeSet="false">
<owner>Owner for Domestic Auto Makers</owner>
    <entityDescription>DomesticAutoMakers</entityDescription>
    <mappings>
        <supportedCodingScheme localId="Automobiles" uri="urn:oid:11.11.0.1">Automobiles</supportedCodingScheme>
        <supportedDataType localId="texthtml">text/html</supportedDataType>
        <supportedDataType localId="textplain">text/plain</supportedDataType>
        <supportedLanguage localId="en" uri="www.en.org/orsomething">en</supportedLanguage>
        <supportedNamespace localId="Automobiles" uri="urn:oid:11.11.0.1" equivalentCodingScheme="Automobiles"
>Automobiles</supportedNamespace>
        <supportedProperty localId="definition">definition</supportedProperty>
        <supportedProperty localId="textualPresentation">textualPresentation</supportedProperty>
        <supportedSource  assemblyRule="rule1" uri="http://informatics.mayo.edu" localId="lexgrid.org">lexgrid.
org</supportedSource>
        <supportedSource localId="_111101">11.11.0.1</supportedSource>
    </mappings>
    <pickListEntryNode pickListEntryId="PL005p1" isActive="true">
        <owner>Owner for PL005p1</owner>
        <entryState containingRevision="R001" relativeOrder="1" changeType="NEW" prevRevision="R00A"/>
        <inclusionEntry entityCode="005" entityCodeNamespace="Automobiles" propertyId="p1">
            <pickText>Domestic Auto Makers</pickText>
            <pickContext>Domestic Auto Makers</pickContext>
            <pickContext>Cars</pickContext>
        </inclusionEntry>
        <properties>
            <property propertyName="definition">
                <entryState containingRevision="R001" relativeOrder="1" changeType="NEW" prevRevision="R00A"/>
                <value dataType="textplain">Definition for Domestic Auto Makers</value>
            </property>
        </properties>
    </pickListEntryNode>
    <pickListEntryNode pickListEntryId="PL005p2" isActive="true">
        <entryState containingRevision="R001" relativeOrder="1" changeType="NEW" prevRevision="R00A"/>
        <inclusionEntry entityCode="005" entityCodeNamespace="Automobiles" propertyId="p2">
            <pickText>American Car Companies</pickText>
        </inclusionEntry>
        <properties>
            <property propertyName="definition">
                <entryState containingRevision="R001" relativeOrder="1" changeType="NEW" prevRevision="R00A"/>
                <value dataType="textplain">Definition for Amerocan Auto Makers</value>
            </property>
        </properties>
    </pickListEntryNode>
</pickListDefinition> </source>
```

Installation / Packaging

Pick List service are integrated parts of core LexEVS API and are packaged and installed with other LexEVS services.

System Testing

The System test case for the LexEVS Pick List Definition service is performed using the JUnit test suite:

`org.LexGrid.LexBIG.Impl.testUtility.VDAllTests`

This test suite will be run as part of regular LexEVS test suites AllTestsAllConfigs and AllTestsNormalConfigs.