LexEVS 6.0 CTS2 Administration 2 - Export Operation API

Contents of this Page

CTS2 Links for LexEVS 6.0

- CTS2 API Main Page
- Programmer's Guide Main Page
- LexEVS 6.0 Main Page
- LexEVS Current Release

Introduction

LexEVS CTS2 Export API provides capability to export complete or partial contents of Code System, Value Sets and Association.

Export Interfaces

There are three major export interfaces proved:

- Code System exporter - Provides capability to export complete or partial contents of Code System.
- Value Sets exporter - Provides capability to export Value Set Definition or Value Set Resolution (Expanded Value Set).
- Association exporter - Provides capability to export Associations.

Each of these interfaces can be accessed using:

```
org.lexevs.cts2.admin.export.CodeSystemExportOperation csExportOp = new org.lexevs.cts2.LexEvsCTS2Impl().
getAdminOperation().getCodeSystemExportOperation();
org.lexevs.cts2.admin.export.ValueSetExportOperation vsExportOp = new org.lexevs.cts2.LexEvsCTS2Impl().
getAdminOperation().getValueSetExportOperation();
org.lexevs.cts2.admin.export.AssociationExportOperation assnExportOp = new org.lexevs.cts2.LexEvsCTS2Impl().
getAdminOperation().getAssociationExportOperation();
```

Code System Exporter

`org.lexevs.cts2.admin.export.CodeSystemExportOperation` is the main interface which can be used to export complete or partial Code System contents. This interface can be accessed using main LexEVSCTS2 interface, like:

```
org.lexevs.cts2.admin.export.CodeSystemExportOperation csExportOp = new org.lexevs.cts2.LexEvsCTS2Impl().
getAdminOperation().getCodeSystemExportOperation();
```

There are three different methods available to export Code System:

- Export complete Code System - This method provides capability to export Code System contents using the exporter specified.
- Export Coded Node Set - This method provides capability to export entities that matches certain restrictions (like matching designation, concept code, property etc).
- Export Coded Node Graph - This method provides capability to export entities that are part of selected graph (like association, source code, target code etc).

Export Complete Code System

This function exports contents of the code system using the exporter specified. By default, LexEVS comes with following exporters:

- LexGrid Exporter - exports contents in LexGrid XML format
- OBO Exporter - exports contents in OBO format
- OWL Exporter - exports contents in OWL/RDF format

`exportCodeSystemContent(String codeSystemNameOrURI, String codeSystemVersion, URI exportDestination, Exporter exporter)`

Description:	Exports contents of the code system using the exporter specified.
Input:	• ***java.lang.String codeSystemNameOrURI*** - (**Mandatory**) Name or URI of the Code System to be exported. • ***java.lang.String codeSystemVersion*** - (Optional) Version of the Code System to be exported. • ***java.net.URI exportDestination*** - (**Mandatory**) Destination path information for the exported file. • ***org.LexGrid.LexBIG.Extensions.Export.Exporter exporter*** - (**Mandatory**) exporter to use. Use getSupportedExporterNames to get all the exporters supported by this instance of CTS2. For example, 'OBOExport' could be used to export code system contents in OBO format, 'OwlRdfExporter' for code system contents in OWL/RDF format, 'LexGridExport' for Code System in LexGrid XML format, etc.
Output:	***java.net.URI*** - URI of destination if successfully exported
Exception:	***org.LexGrid.LexBIG.Exceptions.LBException***

Sample Call:	• *Step 1:* Instantiate CodeSystemExportOperation if it is not done yet:
	`org.lexevs.cts2.admin.export.CodeSystemExportOperation csExportOp = new org.lexevs.cts2.LexEvsCTS2Impl().getAdminOperation().getCodeSystemExportOperation();`
	• *Step 2:* Populate Exporter object to use:
	`LexBIGService lbs = LexBIGServiceImpl.defaultInstance();`
	`org.LexGrid.LexBIG.Impl.exporters.LexGridExport exporter;`
	`try {_`
	`exporter = (org.LexGrid.LexBIG.Impl.exporters.LexGridExport)lbs.getServiceManager(null).getExporter(org.LexGrid.LexBIG.Impl.exporters.LexGridExport.name);`
	`} catch (LBException e) {`
	`throw new RuntimeException(e);`
	`}`
	`org.LexGrid.LexBIG.LexBIGService.CodedNodeGraph cng = lbs.getNodeGraph("Automobiles", Constructors.createCodingSchemeVersionOrTagFromVersion("1.0"), null);`
	`org.LexGrid.LexBIG.LexBIGService.CodedNodeSet cns = lbs.getNodeSet("Automobiles", Constructors.createCodingSchemeVersionOrTagFromVersion("1.0"), null);`
	`exporter.setCng(cng);`
	`exporter.setCns(cns);`
	`exporter.getOptions().getBooleanOption("Async Load").setOptionValue(false);`
	`exporter.getOptions().getBooleanOption(Option.getNameForType(Option.FAIL_ON_ERROR)).setOptionValue(true);`
	`exporter.getOptions().getBooleanOption("force").setOptionValue(true);`
	• *Step 3:* Call export method by passing the code system version, export destination and other parameter values:
	`java.net.URI destURI = csExport.exportCodeSystemContent("Automobiles", "1.0", new File("C:\\").toURI(), exporter);`

Export Coded Node Set

This function resolves the given CodedNodeSet(CNS) and exports the contents in LexGrid XML format. There is a helper method *getCodeSystemCodedNodeSet(String codeSystemNameOrURI, String codeSystemVersion)* that could be used to get the Coded Node Set object for given Code System Version, and could apply further restrictions (ex: matchingDesignation, Status, Properties, Codes etc). And this Coded Node Set object could be supplied to this export method which resolves it and exports the contents in LexGrid XML format.

```
exportCodedNodeSet(String codeSystemNameOrURI, String codeSystemVersion, CodedNodeSet cns, URI exportDestination, boolean overwrite, boolean stopOnErrors, boolean async)
```

Description:	Resolves the given CodedNodeSet(CNS) and exports the contents.
Input:	• ***java.lang.String codeSystemNameOrURI*** - (**Mandatory**) URI of the Code System to be used for resolving the CNS. • ***java.lang.String codeSystemVersion*** - (Optional) Version of the Code System to be exported. • **org.LexGrid.LexBIG.LexBIGService.CodedNodeSet cns** - (**Mandatory**) Coded Node Set object to be resolved and exported. • ***java.net.URI exportDestination*** - (**Mandatory**) Destination path information for the exported file. • ***boolean overwrite*** - (Optional) True means, any existing file will be overwritten. • ***boolean stopOnErrors*** - (Optional) True means stop if any export error is detected. False means attempt to export what can be exported if recoverable errors are encountered. • ***boolean async*** - (Optional) Flag controlling whether export occurs in the calling thread. If true, the export will occur in a separate asynchronous process. If false, this method blocks until the export operation completes or fails. Regardless of setting, the getStatus and getLog calls are used to fetch results.
Output:	***java.net.URI*** - URI of destination if successfully exported
Exception:	***org.LexGrid.LexBIG.Exceptions.LBException***

Sample Call:	• *Step 1:* Instantiate CodeSystemExportOperation if it is not done yet:
	`org.lexevs.cts2.admin.export.CodeSystemExportOperation csExportOp = new org.lexevs.cts2.LexEvsCTS2Impl().getAdminOperation().getCodeSystemExportOperation();`
	• *Step 2:* Populate Coded Node Set object to export and apply restrictions:
	`CodedNodeSet cns = csExport.getCodeSystemCodedNodeSet("Automobiles", "1.0");`
	`cns.restrictToMatchingDesignations("Ford", null, MatchAlgorithms.LuceneQuery.name(), null);`
	• *Step 3:* Call export method by passing the coded node set object , export destination and other parameter values:
	`''java.net.URI destURI = csExport.exportCodedNodeSet("Automobiles", "1.0", cns, new File("c:\\").toURI(), true, true, true);`

Export Coded Node Graph

This function resolves the given CodedNodeGraph(CNG) and exports the contents in LexGrid XML format. There is a helper method *getCodeSystemCodedNodeGraph(String codeSystemNameOrURI, String codeSystemVersion)* that could be used to get the Coded Node Graph object for given Code System Version, and could apply further restrictions (ex: associations, SourceCodes, TargetCodes, etc). And this Coded Node Graph object could be supplied to this export method which resolves it and exports the contents in LexGrid XML format.

```
exportCodedNodeGraph(String codeSystemNameOrURI, String codeSystemVersion, CodedNodeGraph cng, URI exportDestination, boolean overwrite, boolean stopOnErrors, boolean async)
```

Description:	Resolves the given CodedNodeGraph(CNG) and exports the contents.
Input:	• ***java.lang.String codeSystemNameOrURI*** - (**Mandatory**) URI of the Code System to be used for resolving the CNS. • ***java.lang.String codeSystemVersion*** - (Optional) Version of the Code System to be exported. • **''org.LexGrid.LexBIG.LexBIGService.CodedNodeGraph cng** - (**Mandatory**) Coded Node Graph object to be resolved and exported. • ***java.net.URI exportDestination*** - (**Mandatory**) Destination path information for the exported file. • ***boolean overwrite*** - (Optional) True means, any existing file will be overwritten. • ***boolean stopOnErrors*** - (Optional) True means stop if any export error is detected. False means attempt to export what can be exported if recoverable errors are encountered. • ***boolean async*** - (Optional) Flag controlling whether export occurs in the calling thread. If true, the export will occur in a separate asynchronous process. If false, this method blocks until the export operation completes or fails. Regardless of setting, the getStatus and getLog calls are used to fetch results.

Output:	***java.net.URI*** - URI of destination if successfully exported
Exception:	***org.LexGrid.LexBIG.Exceptions.LBException***

Sample Call:	• *Step 1:* Instantiate CodeSystemExportOperation if it is not done yet:
	`org.lexevs.cts2.admin.export.CodeSystemExportOperation csExportOp = new org.lexevs.cts2.LexEvsCTS2Impl().getAdminOperation().getCodeSystemExportOperation();`
	• *Step 2:* Populate Coded Node Graph object to export and apply restrictions:
	`CodedNodeGraph cng = csExport.getCodeSystemCodedNodeGraph("Automobiles", "1.0");`
	`cng.restrictToAssociations(Constructors.createNameAndValueList("uses"), null);`
	• *Step 3:* Call export method by passing the coded node graph object , export destination and other parameter values:
	`''java.net.URI destURI = csExport.exportCodedNodeGraph("Automobiles", "1.0", cng, new File("c:\\").toURI(), true, true, true);`

Value Set Exporter

org.lexevs.cts2.admin.export.ValueSetExportOperation is the main interface which can be used to export Value Set Definition or Value Set Resolution (Expanded Value Set). This interface can be accessed using main LexEVSCTS2 interface, like:

```
org.lexevs.cts2.admin.export.ValueSetExportOperation vsExportOp = new org.lexevs.cts2.LexEvsCTS2Impl().getAdminOperation().getValueSetExportOperation();
```

There are three different methods available to export Value Set:

- Export Value Set Definition - This method provides capability to export Value Set Definition in LexGrid XML format. This will be helpful if there is a need to import this exported Value Set Definition in different instance of LexEVS.
- Export Expanded Value Set using custom exporter - This method provides capability to export contents of Value Set using custom exporter. This will be helpful, if you want the contents exported in format other than LexGrid XML.
- Export Expanded Value Set in LexGrid XML format - This method provides capability to export contents of Value Set using default LexGrid Exporter which exports in LexGrid XML format.

Export Value Set Definition

This method provides capability to export Value Set Definition in LexGrid XML format. This will be helpful if there is a need to import this exported Value Set Definition in different instance of LexEVS.

```
exportValueSetDefinition(URI valueSetDefinitionURI, String valueSetDefinitionVersion, String xmlFullPathName, boolean overwrite, boolean failOnAllErrors)
```

Description:	Export Value Set Definition to LexGrid canonical XML format.
Input:	• ***java.net.URI valueSetDefinitionURI*** - (**Mandatory**) URI of the Value Set Definition to export. • ***java.lang.String valueSetDefinitionVersion*** - (Optional) Version of the Value Set Definition to be exported. • ***java.lang.String xmlFullPathName*** - (**Mandatory**) File location (including file name *.xml) to save the definition. • ***boolean overwrite*** - (Optional) True means, any existing file will be overwritten. • ***boolean failOnAllErrors*** - (Optional) True means stop if any export error is detected. False means attempt to export what can be exported if recoverable errors are encountered.
Output:	None
Exception:	***org.LexGrid.LexBIG.Exceptions.LBException***

Sample Call:	• *Step 1:* Instantiate ValueSetExportOperation if it is not done yet:
	`org.lexevs.cts2.admin.export.ValueSetExportOperation vsExportOp = new org.lexevs.cts2.LexEvsCTS2Impl().getAdminOperation().getValueSetExportOperation();`
	• *Step 2:* Call export method by passing URI of Value Set Definition, output destination and other parameter values:
	`''vsExport.exportValueSetDefinition(new URI("SRITEST:AUTO:PropertyRefTest1-VSDONLY"), null, new File("c:\\TestVSDExport.xml").toURI(), true, true);`

Export Expanded Value Set Using Custom Exporter

This method provides capability to export contents of Value Set using custom exporter. This will be helpful when there is a requirement to export value set contents in format other than LexGrid XML. There is a helper method *getSupportedExporterNames()* that could be used to get all the exporters loaded and supported by current LexEVS instance. So, if you have your custom exporter deployed as an extension to LexEVS Exporter, it will be in the list of exporters returned by method *getSupportedExporterNames()*.

`exportValueSetContents(URI valueSetDefinitionURI, String valueSetDefinitionVersion, URI exportDestination, String exporter)`

Description:	Exports contents of the Value Set Definition using the exporter specified.
Input:	• ***java.lang.String valueSetDefinitionURI*** - (**Mandatory**) URI of the Value Set Definition to be expanded and exported. • ***java.lang.String valueSetDefinitionVersion*** - (Optional) Version of the Value Set Definition to be exported. • ***java.net.URI exportDestination*** - (**Mandatory**) Destination path information for the exported file. • ***java.lang.String exporter*** - (**Mandatory**) Name of the exporter to use. Use getSupportedExporterNames to get all the exporters supported by this instance of CTS2/LexEVS.
Output:	***java.net.URI*** - URI of destination if successfully exported
Exception:	***org.LexGrid.LexBIG.Exceptions.LBException***

Sample Call:	• *Step 1:* Instantiate ValueSetExportOperation if it is not done yet:
	`org.lexevs.cts2.admin.export.ValueSetExportOperation vsExportOp = new org.lexevs.cts2.LexEvsCTS2Impl().getAdminOperation().getValueSetExportOperation();`
	• *Step 2:* Call export method by passing the Value Set Definition URI, custom exporter name, output destination and other parameter values:
	`''java.net.URI destURI = vsExport.exportValueSetContents(new URI("SRITEST:AUTO:PropertyRefTest1-VSDONLY"), null, new File("c://AutomobilesTestVSD.xml").toURI(), "CustomExporter");`

Export Expanded Value Set in LexGrid XML Format

This method provides capability to export contents of Value Set as a ***Code System*** using LexGrid Exporter which exports in LexGrid XML format.

`exportValueSetContents(URI valueSetDefinitionURI, String valueSetDefinitionVersion, URI exportDestination, AbsoluteCodingSchemeVersionReferenceList csVersionList, String csVersionTag, boolean overwrite, boolean failOnAllErrors)`

Description:	Exports contents of Value Set as Code System in LexGrid canonical XML format.

Input:	• ***java.net.URI valueSetDefinitionURI*** - (**Mandatory**) URI of the Value Set Definition to export. • ***java.lang.String valueSetDefinitionVersion*** - (Optional) Version of the Value Set Definition to be exported. • ***java.net.URI exportDestination*** - (**Mandatory**) Location (path to the folder withOUT the file name) to save the definition. ***org.LexGrid.LexBIG.DataModel.Collections.AbsoluteCodingSchemeVersionReferenceList csVersionList*** - (Optional) A list of code system 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 csVersionTag*** - (Optional) The tag (e.g "devel", "production", ...) to be used to reconcile code system when more than one is present. ***boolean overwrite*** - (Optional) True means, any existing file will be overwritten. ***boolean failOnAllErrors*** - (Optional) True means stop if any export error is detected. False means attempt to export what can be exported if recoverable errors are encountered.
Output:	***java.net.URI*** - URI of destination if successfully exported.
Exception:	***org.LexGrid.LexBIG.Exceptions.LBException***

Sample Call:	• *Step 1:* Instantiate ValueSetExportOperation if it is not done yet:
	`org.lexevs.cts2.admin.export.ValueSetExportOperation vsExportOp = new org.lexevs.cts2.LexEvsCTS2Impl().getAdminOperation().getValueSetExportOperation();`
	• *Step 2:* Populate the Code System Version Reference List to be used to resolve the Value Set Definition:
	`AbsoluteCodingSchemeVersionReferenceList csVerList = new AbsoluteCodingSchemeVersionReferenceList(); csVerList.addAbsoluteCodingSchemeVersionReference(Constructors.createAbsoluteCodingSchemeVersionReference("urn:oid:11.11.0.1", "1.0"));`
	• *Step 3:* Call export method by passing URI of Value Set Definition, output destination and other parameter values:
	`''vsExport.exportValueSetDefinition(exportValueSetContents(new URI("SRITEST:AUTO:PropertyRefTest1-VSDONLY"), null, new File("c:\\TestVSDExport.xml").toURI(), csVerList, null, true, true);`

Association Exporter - Export Association

`org.lexevs.cts2.admin.export.AssociationExportOperation` is the main interface which can be used to export Association(s). This interface can be accessed using main LexEVSCTS2 interface, like:

```
org.lexevs.cts2.admin.export.AssociationExportOperation assnExportOp = new org.lexevs.cts2.LexEvsCTS2Impl().getAdminOperation().getAssociationExportOperation();
```

Method **exportAssociation** provides capability to apply filters and export association(s) in LexGrid XML format.

```
exportAssociation(String codeSystemNameOrURI, String codeSystemVersion, CodedNodeGraph cng, URI exportDestination, boolean overwrite, boolean stopOnErrors, boolean async)
```

Description:	Export Association(s) in LexGrid canonical XML format.
Input:	• ***java.lang.String codeSystemNameOrURI*** - (**Mandatory**) Name or URI of the Code System that contains the association. • ***java.lang.String codeSystemVersion*** - (**Mandatory**) Version of the Code System that contains the association. • ***org.LexGrid.LexBIG.LexBIGService.CodedNodeGraph cng*** - (**Mandatory**) Coded Node Graph(CNG) that contains all the filters. This CNG will be resolved and exported. • ***java.net.URI exportDestination*** - (**Mandatory**) Directory location (with out file name). The name of the file will be constructed based on the Code System Name/URI and the Version. • ***boolean overwrite*** - (Optional) True means, any existing file will be overwritten. • ***boolean stopOnErrors*** - (Optional) True means stop if any export error is detected. False means attempt to export what can be exported if recoverable errors are encountered. • ***boolean async*** - (Optional) Flag controlling whether export occurs in the calling thread. If true, the export will occur in a separate asynchronous process. If false, this method blocks until the export operation completes or fails. Regardless of setting, the getStatus and getLog calls are used to fetch results.
Output:	None
Exception:	***org.LexGrid.LexBIG.Exceptions.LBException***

Sample Call:

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

```
org.lexevs.cts2.admin.export.AssociationExportOperation assnExportOp = new org.lexevs.cts2.
LexEvsCTS2Impl().getAdminOperation().getAssociationExportOperation();
```

- *Step 2:* Populate the Coded Node Graph (CNG) and apply filters:

```
org.LexGrid.LexBIG.LexBIGService.CodedNodeGraph cng = LexBIGServiceImpl.defaultInstance().
getNodeGraph("urn:oid:cts:1.1.1",  "1.0", null);_
org.LexGrid.LexBIG.DataModel.Core.NameAndValue nv = new org.LexGrid.LexBIG.DataModel.Core.
NameAndValue();_
nv.setName("hasSubtype");_
org.LexGrid.LexBIG.DataModel.Collections.NameAndValueList assocNames = new org.LexGrid.LexBIG.
DataModel.Collections.NameAndValueList();_
assocNames.addNameAndValue(nv);_
cng.restrictToAssociations(assocNames, null);_
```

- *Step 3:* Call export method by passing Code System Information, Coded Node Graph, destination path and other parameter values:

```
''assnExport.exportAssociation("urn:oid:cts:1.1.1",  "1.0", cng, new File("c:
\\exportedFiles\\").toURI(), true, true, true);
```

Exporter Mappings - OwlRdf Exporter Mapping

OwlRdf exporter is based on Jena 2.6.3. It use SDB to build up triple store. The triple store tables are under the same database of LexEvs. LexEvs retrieval api does not support to fetch AssociatedData. It cannot retrieve the association that is from an entity
to a data/value. The owl/rdf exporter, based on LexEVS has the limiation of handling the owl:hasValue, owl:maxCardinality, owl:minCardinality, owl:cardinality constraints as well.

LexGrid	**Owl/Rdf**	**Comment**
CodingScheme		
codingScheme	owl:Ontology	
codingScheme.source	dc:source	a property of owl:ontology
codingScheme.copyright	dc:right	a property of owl:ontology
codingScheme.Name	rdfs:label	a property of owl:ontology
codingScheme.URI	xmlns	a property of owl:ontology
codingScheme.representVersion	owl:versionInfo	a property of owl:ontology
codingScheme.formalName	dc:title	a property of owl:ontology
codingScheme.defaultLanguage	dc:language	a property of owl:ontology
codingScheme.approxNumConcpts	not mapped	
Entity		
entity	skos:Concept	
concept	owl:Class	
instance entity	owl:Thing	
associationEntity	owl:property	owl:objectProperty or owl:datatypeProperty
entityCode	rdf:ID	local name
entityCodeNamespace	xmlns	
isDefined	LexRDF:isDefined	LexRDF is a namespace generated by Mayo
Property		**Use reification statement for the property's property**

property	owl:AnnotationProperty when no type specified	
language	dc:language	
source	dc:source	
comment	skos:note	
presentation	skos:altLabel, skos:prefLabel	According to lg isPrefered
definition	skos:definition	
isPreferred	LexRDF:isPreferred	
degreeOfFidelity	LexRDF:degreeOfFidelity	
matchIfNoContext	LexRDF:matchIfNoContext	
representationForm	LexRDF:presentationForm	
propertyLink	LexRDF:propertyLink	
Association		
associationPredicate. associationName	rdf:id	
associationEntity	owl:property	owl:objectProperty or owl:datatypeProperty
associationEntity.forwardName	rdf:id	if not null
associationEntity.isTransitive	owl:TransitiveProperty	