LexEVS 4.2 Grid Service User Guide

The information and links on this page are no longer being updated and are provided for reference purposes only.

Contents of this Page

Overview

The cancer Biomedical Informatics Grid, or caBIG®, is a virtual informatics infrastructure that connects data, research tools, scientists, and organizations to leverage their combined strengths and expertise in an open environment with common standards and shared tools. The current test bed architecture of caBIG® is dubbed caGrid. The software embodiment and corresponding documentation of this architecture constitute the caGrid 1.2 release.

This User Guide addresses the LexEVS Grid Service (version 4.2), from the perspective of a client application developer looking to interface with the referenced analytical grid service.

Note

The current version of the LexEVS Grid Service is 4.2 and it interfaces with the EVS API 4.2.

Release History

Version Number	Date	Description
1.0	Unknown	Original release of the EVS Grid Service. Interfaced with caCORE/EVS 3.1.
4.1	June 27, 2008	Follow-on release of EVS Grid Service which interfaces with EVS API 4.1 (based on the EVS 3.2 OM).
4.2	October 11, 2008	Follow-on release of EVS Grid Service 4.1 built on top of LexBIG server 2.3 which interfaces with EVS API 4.2

The LexEVS Grid Service

The following table summarizes the operations available through the LexEVS Analytical Grid Service. Each of the operations is also defined in detail below. The grid analytical service and related operations are viewable via the caGrid Portal (http://cagrid-portal.nci.nih.gov).

Using the API

There are two (2) different interfaces for accessing the LexEVS Grid Services:

1. `org.LexGrid.LexBIG.cagrid.adapters.LexBIGServiceAdapter` or
2. `org.LexGrid.LexBIG.cagrid.adapters.LexBIGServiceGridAdapter`

- **Option 1, org.LexGrid.LexBIG.cagrid.adapters.LexBIGServiceAdapter** provides an interface for interacting with the LexEVS Grid Services. This Interface is intended to mirror the existing LexBIG API as much as possible. There is no object wrapping for semantic purposes on this interface. This allows existing applications of the LexBIG API to use Grid Services without code changes.
  This Interface may be acquired by instantiating LexBIGServiceAdapter with the Grid Service URL as a parameter.

```
LexBIGService lbs = new LexBIGServiceAdapter("http://lexevsapi.nci.nih.gov/wsrf/services/cagrid
/LexEVSGridService");
```

- **Option 2, org.LexGrid.LexBIG.cagrid.adapters.LexBIGServiceGridAdapter** also provides an interface for interacting with the LexEVS Grid Services. However, this Interfaces is the semantically defined interface. All method parameters and return values are defined and annotated as CDEs to be loaded into caDSR. This Interface is intended to be caGrid Silver Level Compliant.
  This Interface may be acquired by instantiating LexBIGServiceGridAdapter with the Grid Service URL as a parameter.

```
LexBIGServiceGrid lbs = new LexBIGServiceGridAdapter("http://lexevsapi.nci.nih.gov/wsrf/services/cagrid
/LexEVSGridService);
```

Method Descriptions

getCodingSchemeConcepts

getCodingSchemeConcepts (CodingSchemeIdentification, CodingSchemeVersionOrTag)	–
Description:	Returns the set of all (or all active) concepts in the specified coding scheme.
Input:	*org.LexGrid.LexBIG.DataModel.cagrid.CodingSchemeIdentification, org.LexGrid.LexBIG.DataModel.Core. CodingSchemeVersionOrTag*
Output:	*org.LexGrid.LexBIG.cagrid.LexBIGCaGridServices.CodedNodeSet.stubs.types.CodedNodeSetReference*
Exception:	*RemoteException*
Implementation Details:	***Implementation:*** • *Step 1*: Create a Resource on the server and populate it with the requested `org.LexGrid.LexBIG.LexBIGService.CodedNodeSet`. • *Step 2*: Return the Client Reference to the user. This Reference has the above `org.LexGrid.LexBIG.LexBIGService.CodedNodeSet` as a Resource. An `org.LexGrid.LexBIG.cagrid.LexBIGCaGridServices.service.CodedNodeSetClient` object is built from the above Reference.

Sample Call	***Sample Call*** • *Step 1*: Connect to the LexBIG caGrid Service using the `org.LexGrid.LexBIG.cagrid.adapters.LexBIGServiceAdapter` or `org.LexGrid.LexBIG.cagrid.adapters.LexBIGServiceGridAdapter` `LexBIGServiceGrid lbs = new LexBIGServiceGridAdapter(url);` • *Step 2*: Build a `org.LexGrid.LexBIG.DataModel.Core.CodingSchemeVersionOrTag` containing the Version information for the desired Coding Scheme `CodingSchemeVersionOrTag csvt = new CodingSchemeVersionOrTag(); csvt.setVersion ("testVersion");` • *Step 3*: Build an `org.LexGrid.LexBIG.DataModel.cagrid.CodingSchemeIdentification` to hold the Coding Scheme name. `CodingSchemeIdentification codingScheme = new CodingSchemeIdentification(); codingScheme.setCode(code);` • *Step 4*: Invoke the LexBIG caGrid service as follows: `CodedNodeSetGrid cns = lbs. getCodingSchemeConcepts(codingScheme, csvt);`

getFilter

getFilter (ExtensionIdentification)	–
Description:	Returns an instance of the filter extension registered with the given name.
Input:	*org.LexGrid.LexBIG.DataModel.cagrid.ExtensionIdentification*
Output:	*org.LexGrid.LexBIG.cagrid.LexBIGCaGridServices.Filter.stubs.types.FilterReference*
Exception:	*RemoteException*
Implementation Details:	***Implementation:*** • *Step 1*: Create a Resource on the server and populate it with the requested `org.LexGrid.LexBIG.Extensions. Query.Filter`. • *Step 2*: Return the Client Reference to the user. This Reference has the above `org.LexGrid.LexBIG. Extensions.Query.Filter` as a Resource. This client is a Service Context that allows the user to call regular org.LexGrid.LexBIG.Extensions.Query.Filter API calls through the grid service. An `org.LexGrid.LexBIG. cagrid.LexBIGCaGridServices`.Filter.client.FilterClient}} object is built from the above Reference. This FilterClient implements the Interface org.LexGrid.LexBIG.Extensions.Query.Filter. This makes calling Grid Service Calls through `org.LexGrid.LexBIG.cagrid.LexBIGCaGridServices.Filter.client.FilterClient` transparent to the end user.
Sample Call	***Sample Call:*** • *Step 1*: Connect to the LexBIG caGrid Service using the `org.LexGrid.LexBIG.cagrid.adapters. LexBIGServiceAdapter` or `org.LexGrid.LexBIG.cagrid.adapters.LexBIGServiceGridAdapter` `LexBIGServiceGrid lbs = new LexBIGServiceGridAdapter(url);` • *Step 2*: Build an `org.LexGrid.LexBIG.DataModel.cagrid.ExtensionIdentification` to hold the Extension name. `ExtensionIdentification extension = new ExtensionIdentification(); extension. setLexBIGExtensionName(name);` • *Step 3*: Invoke the LexBIG caGrid service as follows: `Filter filter = lbs.getFilter(extension);`

getSortAlgorithm

getSortAlgorithm (ExtensionIdentification)	–
Description:	Returns an instance of the sort extension registered with the given name.
Input:	*org.LexGrid.LexBIG.DataModel.cagrid.ExtensionIdentification*
Output:	*org.LexGrid.LexBIG.cagrid.LexBIGCaGridServices.Sort* stubs.types.SortReference
Exception:	*RemoteException*

Implementation Details:	***Implementation*** : • *Step 1*: Create a Resource on the server and populate it with the requested `org.LexGrid.LexBIG.Extensions.Query.Sort` • *Step 2*: Return the Client Reference to the user. This Reference has the above org.LexGrid.LexBIG.cagrid.LexBIGCaGridServices.Sort.client.SortClient as a Resource. This client is a Service Context that allows the user to call regular `org.LexGrid.LexBIG.Extensions.Query.Sort` API calls through the grid service. An `org.LexGrid.LexBIG.cagrid.LexBIGCaGridServices.Sort.client.SortClient` object is built from the above Reference. This SortClient implements the Interface org.LexGrid.LexBIG.Extensions.Query.Sort. This makes calling Grid Service Calls through `org.LexGrid.LexBIG.cagrid.LexBIGCaGridServices.Sort.client.SortClient` transparent to the end user.

Sample Call	***Sample Call*** :
	• *Step 1*: Connect to the LexBIG caGrid Service using the `org.LexGrid.LexBIG.cagrid.adapters.LexBIGServiceAdapter` or `org.LexGrid.LexBIG.cagrid.adapters.LexBIGServiceGridAdapter`
	`LexBIGServiceGrid lbs = new LexBIGServiceGridAdapter(url);`
	• *Step 2*: Build an `org.LexGrid.LexBIG.DataModel.cagrid.ExtensionIdentification` to hold the Extension name.
	`ExtensionIdentification extension = new ExtensionIdentification(); extension.setLexBIGExtensionName(name);`
	• *Step 3*: Invoke the LexBIG caGrid service as follows:
	`Filter filter = lbs.getSortAlgorithm(extension);`

getFilterExtensions

getFilterExtensions()	–
Description:	Returns a description of all registered extensions used to provide additional filtering of query results.
Input:	*none*
Output	*org.LexGrid.LexBIG.DataModel.Collections.ExtensionDescriptionList*
Exception:	*RemoteException*
Implementation Details:	***Implementation*** : *Step 1:* Call this method on the associated LexBIG Service instance (or Distributed LexBIG instance) on the server, and forward the results.

Sample Call	***Sample Call*** :
	• *Step 1:* Connect to the LexBIG caGrid Service using the `org.LexGrid.LexBIG.cagrid.adapters.LexBIGServiceAdapter` or `org.LexGrid.LexBIG.cagrid.adapters.LexBIGServiceGridAdapter`
	`LexBIGServiceGrid lbs = new LexBIGServiceGridAdapter(url);`
	• *Step 2:* Invoke the LexBIG caGrid service as follows:
	`ExtensionDescriptionList extDescList = lbs.getFilterExtensions();`

getServiceMetadata

getServiceMetadata()	–
Description:	Return an interface to perform system-wide query over metadata for loaded code systems and providers.
Input:	*none*
Output:	*org.LexGrid.LexBIG.cagrid.LexBIGCaGridServices.LexBIGServiceMetadata.* *stubs.types.LexBIGServiceMetadataReference*
Exception:	*RemoteException*
Implementation Details:	***Implementation:*** • *Step 1:* Create a Resource on the server and populate it with the requested `org.LexGrid.LexBIG.LexBIGService.LexBIGServiceMetadata`. • *Step 2:* Return the LexBIGServiceMetadataClient to the user. This LexBIGServiceMetadataClient has the above `org.LexGrid.LexBIG.LexBIGService.LexBIGServiceMetadata` as a Resource. An `org.LexGrid.LexBIG.cagrid.LexBIGCaGridServices.service.LexBIGServiceMetadataClient` object is built from the above Reference.

Sample Call	***Sample Call:***
	• *Step 1:* Connect to the LexBIG caGrid Service using the `org.LexGrid.LexBIG.cagrid.adapters.LexBIGServiceAdapter` or `org.LexGrid.LexBIG.cagrid.adapters.LexBIGServiceGridAdapter`
	`LexBIGServiceGrid lbs = new LexBIGServiceGridAdapter(url);`
	• *Step 2:* Invoke the LexBIG caGrid service as follows:
	`LexBIGServiceMetadataGrid metadata = lbs.getServiceMetadata();`

getSupportedCodingSchemes

getSupportedCodingSchemes()	–
Description:	Return a list of coding schemes and versions that are supported by this service, along with their status.
Input:	*none*
Output:	*org.LexGrid.LexBIG.DataModel.Collections.CodingSchemeRenderingList*
Exception:	*RemoteException*
Implementation Details:	***Implementation:*** *Step 1:* Call this method on the associated LexBIG Service instance (or Distributed LexBIG instance) on the server, and forward the results.

Sample Call	***Sample Call:***
	• *Step 1:* Connect to the LexBIG caGrid Service using the `org.LexGrid.LexBIG.cagrid.adapters.LexBIGServiceAdapter` or `org.LexGrid.LexBIG.cagrid.adapters.LexBIGServiceGridAdapter`
	`LexBIGServiceGrid lbs = new LexBIGServiceGridAdapter(url);`
	• *Step 2:* Invoke the LexBIG caGrid service as follows:
	`CodingSchemeRenderingList csrl = lbs.getSupportedCodingSchemes();`

getLastUpdateTime

getLastUpdateTime()	–
Description:	Return the last time that the content of this service was changed; null if no changes have occurred. Tag assignments do not count as service changes for this purpose.
Input:	*none*
Output:	*java.util.Date*
Exception:	*RemoteException*
Implementation Details:	***Implementation:*** *Step 1:* Call this method on the associated LexBIG Service instance (or Distributed LexBIG instance) on the server, and forward the results.

Sample Call	***Sample Call:***
	• *Step 1:* Connect to the LexBIG caGrid Service using the `org.LexGrid.LexBIG.cagrid.adapters.LexBIGServiceAdapter` or `org.LexGrid.LexBIG.cagrid.adapters.LexBIGServiceGridAdapter`.
	`LexBIGServiceGrid lbs = new LexBIGServiceGridAdapter(url);`
	• *Step 2:* Invoke the LexBIG caGrid service as follows:
	`Date date = lbs.getLastUpdateTime();`

resolveCodingScheme

resolveCodingScheme (CodingSchemeIdentification, CodingSchemeVersionOrTag)	–
Description:	Return detailed coding scheme information given a specific tag or version identifier.
Input:	*org.LexGrid.LexBIG.DataModel.cagrid.CodingSchemeIdentification, org.LexGrid.LexBIG.DataModel.Core.CodingSchemeVersionOrTag*

Output:	*org.LexGrid.codingSchemes.CodingScheme*
Exception:	*RemoteException*

Implementation Details:	***Implementation:***
	Step 1: Call this method on the associated LexBIG Service instance (or Distributed LexBIG instance) on the server, and forward the results.
	Sample Call:
	• *Step 1*: Connect to the LexBIG caGrid Service using the `org.LexGrid.LexBIG.cagrid.adapters.LexBIGServiceAdapter` or org.LexGrid.LexBIG.cagrid.adapters.LexBIGServiceGridAdapter
	`LexBIGServiceGrid lbs = new LexBIGServiceGridAdapter(url);`
	• *Step 2*: Build an `org.LexGrid.LexBIG.DataModel.cagrid.CodingSchemeIdentification` to hold the Coding Scheme name.
	`CodingSchemeIdentification codingScheme = new CodingSchemeIdentification(); codingScheme.setCode(code);`
	• *Step 3*: Build a `org.LexGrid.LexBIG.DataModel.Core.CodingSchemeVersionOrTag` containing the Version information for the desired Coding Scheme
	`CodingSchemeVersionOrTag csvt = new CodingSchemeVersionOrTag(); csvt.setVersion("testVersion");`
	• *Step 4*: Invoke the LexBIG caGrid service as follows: `CodedNodeSetGrid cns = lbs.resolveCodingScheme(codingScheme, csvt);`

getNodeGraph

getNodeGraph (CodingSchemeIdentification, CodingSchemeVersionOrTag, RelationContainerIdentification)	–
Description:	Returns the node graph as represented in the particular relationship set in the coding scheme.
Input:	*org.LexGrid.LexBIG.DataModel.cagrid.CodingSchemeIdentification, org.LexGrid.LexBIG.DataModel.Core.CodingSchemeVersionOrTag, org.LexGrid.LexBIG.DataModel.cagrid.RelationContainerIdentification*
Output:	*org.LexGrid.LexBIG.cagrid.LexBIGCaGridServices.* CodedNodeGraph.stubs.types.CodedNodeGraphReference"
Exception:	*RemoteException*
Implementation Details:	***Implementation*** :* • *Step 1*: Create a Resource on the server and populate it with the requested `org.LexGrid.LexBIG.LexBIGService.CodedNodeGraph`. • *Step 2*: Return the Client Reference to the user. This Reference has the above `org.LexGrid.LexBIG.LexBIGService.CodedNodeGraph` as a Resource. An `org.LexGrid.LexBIG.cagrid.LexBIGCaGridServices.service.CodedNodeGraphClient` object is built from the above Reference.

Sample Call	***Sample Call*** :
	• *Step 1*: Connect to the LexBIG caGrid Service using the `org.LexGrid.LexBIG.cagrid.adapters.LexBIGServiceAdapter` or `org.LexGrid.LexBIG.cagrid.adapters.LexBIGServiceGridAdapter`
	`LexBIGServiceGrid lbs = new LexBIGServiceGridAdapter(url);`
	• *Step 2*: Build an `org.LexGrid.LexBIG.DataModel.cagrid.CodingSchemeIdentification` to hold the Coding Scheme name.
	`CodingSchemeIdentification codingScheme = new CodingSchemeIdentification(); codingScheme.setCode(code);`
	• *Step 3*: Build an `org.LexGrid.LexBIG.DataModel.Core.CodingSchemeVersionOrTag` containing the Version information for the desired Coding Scheme
	`CodingSchemeVersionOrTag csvt = new CodingSchemeVersionOrTag(); csvt.setVersion("testVersion");`
	• *Step 4*: Build an `org.LexGrid.LexBIG.DataModel.cagrid.RelationContainerIdentification` containing the Relation Container information.
	`RelationContainerIdentification container = new RelationContainerIdentification(); container.setDc(name);`
	• *Step 5*: Invoke the LexBIG caGrid service as follows, providing String parameters for the desired Coding Scheme and Relationship Name: `CodedNodeGraphGrid cng = client.getNodeGraph (codingScheme, csvt, container);`

getMatchAlgorithms

getMatchAlgorithms()	–
Description:	Returns the node graph as represented in the particular relationship set in the coding scheme.
Input:	*none*
Output:	*org.LexGrid.LexBIG.DataModel.Collections.ModuleDescriptionList*
Exception:	*RemoteException*
Implementation Details:	***Implementation:*** *Step 1:* Call this method on the associated LexBIG Service instance (or Distributed LexBIG instance) on the server, and forward the results.

Sample Call	***Sample Call:***
	• *Step 1:* Connect to the LexBIG caGrid Service using the `org.LexGrid.LexBIG.cagrid.adapters.LexBIGServiceAdapter` or `org.LexGrid.LexBIG.cagrid.adapters.LexBIGServiceGridAdapter`
	`LexBIGServiceGrid lbs = new LexBIGServiceGridAdapter(url);`
	• *Step 2:* Invoke the LexBIG caGrid service as follows:
	`ModuleDescriptionList mdl = lbs.getMatchAlgorithms();`

getGenericExtensions

getGenericExtensions()	–
Description:	Returns a description of all registered extensions used to implement application-specific behavior that is centrally accessible from a LexBIGService. **Note** Only generic extensions (base class GenericExtension) will be listed here. All other classes are retrievable at the appropriate interface point (filter, sort, etc).
Input:	*none*
Output:	*org.LexGrid.LexBIG.DataModel.Collections.ExtensionDescriptionList*
Exception:	*RemoteException*

Implementation Details:	***Implementation:***
	Step 1: Call this method on the associated LexBIG Service instance (or Distributed LexBIG instance) on the server, and forward the results.
	Sample Call:
	• *Step 1:* Connect to the LexBIG caGrid Service using the `org.LexGrid.LexBIG.cagrid.adapters.LexBIGServiceAdapter` or `org.LexGrid.LexBIG.cagrid.adapters.LexBIGServiceGridAdapter`
	`LexBIGServiceGrid lbs = new LexBIGServiceGridAdapter(url);`
	• *Step 2:* Invoke the LexBIG caGrid service as follows:
	`ExtensionDescriptionList edl = lbs.getGenericExtensions();`

getGenericExtension

getGenericExtensions (ExtensionIdentification)	–
Description:	Returns an instance of the application-specific extension registered with the given name.
Input:	*org.LexGrid.LexBIG.DataModel.cagrid.ExtensionIdentification*
Output:	*org.LexGrid.LexBIG.DataModel.Collections.SortDescriptionList*
Exception:	*RemoteException*

Implementation Details:	***Implementation*** :
	Step 1: Call this method on the associated LexBIG Service instance (or Distributed LexBIG instance) on the server, and forward the results.
	Sample Call :
	• *Step 1*: Connect to the LexBIG caGrid Service using the `org.LexGrid.LexBIG.cagrid.adapters.LexBIGServiceAdapter` or `org.LexGrid.LexBIG.cagrid.adapters.LexBIGServiceGridAdapter`
	`LexBIGServiceGrid lbs = new LexBIGServiceGridAdapter(url);`
	Note
	Currently this method will return a LexBIGServiceConvenienceMethods instance.
	• *Step 2*: Build an `org.LexGrid.LexBIG.DataModel.cagrid.ExtensionIdentification` to hold the Extension name. `ExtensionIdentification extension = new ExtensionIdentification();`
	`extension.setLexBIGExtensionName("LexBIGServiceConvenienceMethods");`
	• *Step 3*: Invoke the LexBIG caGrid service as follows: `LexBIGServiceConvenienceMethodsGrid lbscm = lbs.getGenericExtensions(extension);`
	• *Step 4*: Return the `LexBIGServiceConvenienceMethodsClient` to the user. This LexBIGServiceConvenienceMethodsClient has the above `org.LexGrid.LexBIG.Extensions.Generic.LexBIGServiceConvenienceMethods` as a Resource. An `org.LexGrid.LexBIG.cagrid.LexBIGCaGridServices.service.CodedNodeGraphClient` object is built from the above Reference.

getHistoryService

getHistoryService (CodingSchemeIdentification)	–
Description:	Resolve a reference to the history api servicing the given coding scheme.
Input:	*org.LexGrid.LexBIG.DataModel.cagrid.CodingSchemeIdentification*
Output:	*org.LexGrid.LexBIG.cagrid.LexBIGCaGridServices.HistoryService.stubs.types.HistoryServiceReference*
Exception:	*RemoteException*
Implementation Details:	***Implementation*** : • *Step 1*: Call this method on the associated LexBIG Service instance (or Distributed LexBIG instance) on the server, and forward the results. • *Step 2*: Return the HistoryServiceClient to the user. This HistoryServiceClient has the above `org.LexGrid.LexBIG.History.HistoryService` as a Resource. This Client is a Service Context that allows the user to call regular `org.LexGrid.LexBIG.History.HistoryService` API calls through the grid service. HistoryServiceClient implements the Interface `org.LexGrid.LexBIG.History.HistoryService`. This makes calling Grid Service Calls through `org.LexGrid.LexBIG.cagrid.LexBIGCaGridServices.HistoryService.client.HistoryServiceClient` transparent to the end user.

Sample Call	***Sample Call*** :
	• *Step 1*: Connect to the LexBIG caGrid Service using the `org.LexGrid.LexBIG.cagrid.adapters.LexBIGServiceAdapter` or `org.LexGrid.LexBIG.cagrid.adapters.LexBIGServiceGridAdapter`
	`LexBIGServiceGrid lbs = new LexBIGServiceGridAdapter(url);`
	• *Step 2*: Build an `org.LexGrid.LexBIG.DataModel.cagrid.CodingSchemeIdentification` to hold the Coding Scheme name.
	`CodingSchemeIdentification codingScheme = new CodingSchemeIdentification();`
	`codingScheme.setCode(code);`
	• *Step 3*: Invoke the LexBIG caGrid service as follows: `HistoryServiceGrid history = lbs.getHistoryService(codingScheme);`

getSortAlgorithms

getSortAlgorithms (SortContext)	–
Description:	Returns a description of all registered extensions used to provide additional filtering of query results.
Input:	*org.LexGrid.LexBIG.DataModel.InterfaceElements.types.SortContext*

Output:	*org.LexGrid.LexBIG.DataModel.Collections.SortDescriptionList*
Exception:	*RemoteException*
Implementation Details:	***Implementation*** : *Step 1*: Call this method on the associated LexBIG Service instance (or Distributed LexBIG instance) on the server, and forward the results. ***Sample Call*** : • *Step 1*: Connect to the LexBIG caGrid Service using the `org.LexGrid.LexBIG.cagrid.adapters.LexBIGServiceAdapter` or `org.LexGrid.LexBIG.cagrid.adapters.LexBIGServiceGridAdapter` `LexBIGServiceGrid lbs = new LexBIGServiceGridAdapter(url);` • *Step 2*: Invoke the LexBIG caGrid service as follows: `SortDescriptionList sortDescList = lbs.getSortAlgorithms(sortContext);`

resolveCodingSchemeCopyright

resolveCodingSchemeCopyright (CodingSchemeIdentification)	–
Description:	Return coding scheme copyright given a specific tag or version identifier.
Input:	*org.LexGrid.LexBIG.DataModel.cagrid.CodingSchemeIdentification*
Output:	*org.LexGrid.LexBIG.DataModel.cagrid.CodingSchemeCopyRight*
Exception:	*RemoteException*

Implementation Details:	***Implementation*** :
	Step 1: Call this method on the associated LexBIG Service instance (or Distributed LexBIG instance) on the server}}, and forward the results.
	Sample Call :
	• *Step 1*: Connect to the LexBIG caGrid Service using the `org.LexGrid.LexBIG.cagrid.adapters.LexBIGServiceAdapter` or `org.LexGrid.LexBIG.cagrid.adapters.LexBIGServiceGridAdapter` `LexBIGServiceGrid lbs = new LexBIGServiceGridAdapter(url);`
	• *Step 2*: Build an `org.LexGrid.LexBIG.DataModel.cagrid.CodingSchemeIdentification` to hold the Coding Scheme name.
	`CodingSchemeIdentification codingScheme = new`
	`CodingSchemeIdentification();`
	`codingScheme.setCode;`
	• *Step 3*: Build an `org.LexGrid.LexBIG.DataModel.Core.CodingSchemeVersionOrTag` containing the Version information for the desired Coding Scheme
	`CodingSchemeVersionOrTag csvt = new CodingSchemeVersionOrTag(); csvt.setVersion("testVersion");`
	• *Step 4*: Invoke the LexBIG caGrid service as follows: `CodingSchemeCopyRight copyright = lbs.resolveCodingSchemeCopyright(codingScheme, csvt);`

setSecurityToken

setSecurityToken (CodingSchemeIdentification, SecurityToken)	–
Description:	Sets the Security Token for the given Coding Scheme.
Input:	*org.LexGrid.LexBIG.DataModel.cagrid.CodingSchemeIdentification, gov.nih.nci.evs.security.SecurityToken*
Output:	*org.LexGrid.LexBIG.cagrid.LexEVSGridService.stubs.types.LexEVSGridServiceReference.LexEVSGridServiceReference*
Exception:	*RemoteException*

Implementation Details:	***Implementation*** :
	Step 1: Call this method on the associated LexBIG Service instance (or Distributed LexBIG instance) on the server, and forward the results.
	Sample Call :
	• *Step 1*: Connect to the LexBIG caGrid Service using the `org.LexGrid.LexBIG.cagrid.adapters.LexBIGServiceAdapter` or `org.LexGrid.LexBIG.cagrid.adapters.LexBIGServiceGridAdapter`.
	LexBIGServiceGrid lbs = new LexBIGServiceGridAdapter(url);
	• *Step 2*: Build an `org.LexGrid.LexBIG.DataModel.cagrid.CodingSchemeIdentification` to hold the Coding Scheme name.
	`CodingSchemeIdentification codingScheme = new CodingSchemeIdentification(); codingScheme.setName("codingScheme");`
	• *Step 3*: Build an `gov.nih.nci.evs.security.SecurityToken` containing the security information for the desired Coding Scheme.
	`SecurityToken metaToken = new SecurityToken(); metaToken.setAccessToken("token");`
	• *Step 4*: Invoke the LexBIG caGrid service as follows: This will return a reference to a new "LexBIGServiceGrid" instance that is associated with the security properties that were passed in.
	`LexBIGServiceGrid lbsg = lbs.setSecurityToken(codingScheme, metaToken);`

Usage Instructions

Service URL

The LexEVS Grid Service 4.2 URL is:

```
Link provided for historical purposes http://lexevsapi.nci.nih.gov/wsrf/services/cagrid/LexEVSGridService
```

The service is also accessible via the caGRID Portal.

Required Libraries

The libraries required for programmatic access to the LexEVS Grid Service are listed in the following tables.

The table below is the Third-Party Software Libraries required for use of the LexEVS API Grid Service.

Product	Jars	License	Home Page
Apache WS-Addressing	addressing-1.0.jar	adressing 1.0.LICENSE	From Globus 4.0.2 Java Web Services Core lib directory: http://www.globus.org/toolkit/downloads/4.0.2 Source available at: `Link provided for historical purposes http://ws.apache.org/addressing`
Apache Axis	• axis-ant.jar • axis.jar • commons-pool-1.3.jar • commons-logging-1.1.jar • commons-lang-2.2.jar • commons-collections-3.2.jar • commons-codec-1.3.jar • log4j-1.2.8.jar • jaxrpc.jar • saaj.jar • wsdl4j.jar	axis-jars.LICENSE	http://ws.apache.org/axis
Apache Xerces	xercesImpl.jar	xerces.LICENSE	http://xerces.apache.org/xerces-j

Apache Lucene	• lucene-core-2.3.2.jar • lucene-regex-2.3.2.jar • lucene-snowball-2.3.2.jar	Lucene LICENSE	http://lucene.apache.org/
ASM - all purpose Java bytecode manipulation and analysis framework	`asm.jar`	http://asm.objectweb.org/license.html	http://asm.objectweb.org/
Castor	castor-1.2.jar	http://www.castor.org/license.html	`Link provided for historical purposes http://www.castor.org/index.html`
Globus Toolkit	• cog-axis.jar • cog-jglobus.jar	http://www.globus.org/toolkit/legal/4.0/	–
Bouncy Castle Crypto APIs	jce-jdk13-125.jar	http://www.bouncycastle.org/licence.html	http://www.bouncycastle.org/
Open Permis	• wsrf_core.jar • wsrf_core_stubs.jar	http://www.openpermis.org/BSDlicenceKent.txt	http://www.openpermis.org/
Apache WSS4J	wss4j.jar	http://ws.apache.org/wss4j/license.html	http://ws.apache.org/wss4j/
Spring	spring.jar	Spring LICENSE	http://www.springframework.org

The table below lists the NCICB software captured under the caBIG® umbrella.

Library	Associated JARs
caGrid Software Libraries	caGrid-ServiceSecurityProvider-client-1.2.jar
caGrid Software Libraries	caGrid-ServiceSecurityProvider-common-1.2.jar
caGrid Software Libraries	caGrid-ServiceSecurityProvider-stubs-1.2.jar
caGrid Software Libraries	caGrid-core-1.2.jar
caGrid Software Libraries	caGrid-metadata-common-1.2.jar
caGrid Software Libraries	caGrid-metadata-data-1.2.jar
caGrid Software Libraries	caGrid-metadata-security-1.2.jar
caGrid Software Libraries	caGrid-metadatautils-1.2.jar
EVS API Libaries	evsapi42-beans.jar
EVS API Libaries	evsapi42-framework.jar
LexEVS Grid Service Client Library	LexEVSGridService-client.jar
LexEVS Grid Service Stubs	LexEVSGridService-stubs.jar
LexEVS Grid Service Common	LexEVSGridService-common.jar
LexEVS Grid Service Service	LexEVSGridService-service.jar
LexEVS Grid Service Tests	LexEVSGridService-tests.jar
caCORE SDK Library	sdk-client-framework.jar
LexBIG API	lexbig.jar
Custom Castor Serializer	castor-bean-serializer.jar

Downloads

For your convenience, the required libraries are available for download on GForge

In order to programmatically access the LexEVS API Grid Service, these libraries need to be added to your local classpath.

Code Examples

For an example client, service calls, and SOAP messages, downoload the TestClient zip file.

Example API Usage

Searching for concepts in NCI Thesaurus containing the string "Gene"

```
//Create a Connection to the Grid Service
LexBIGServiceGrid lbs = new LexBIGServiceGridAdapter(gridServiceURL);

//Set up the CodingSchemeIdentification object to define the Coding Scheme
CodingSchemeIdentification csid = new CodingSchemeIdentification();
csid.setName("NCI Thesaurus");

//Get the CodedNodeSet for that CodingScheme (This returns a CodedNodeSet Service Context)
CodedNodeSetGrid cnsg = lbs.getCodingSchemeConcepts(csid, null); //getCodingSchemeConcepts is a Grid Service
Call

//Set the text to match
MatchCriteria matchText = new MatchCriteria();
matchText.setText("Gene");

//Define a SearchDesignationOption, if any
SearchDesignationOption searchOption = new SearchDesignationOption();

//Choose an algorithm to do the matching
ExtensionIdentification matchAlgorithm = new ExtensionIdentification();
matchAlgorithm.setLexBIGExtensionName("contains");

//Chose a language
LanguageIdentification language = new LanguageIdentification();
language.setIdentifier("en");

//Restrict the CodedNodeSet
cnsg.restrictToMatchingDesignations(matchText, searchOption, matchAlgorithm, language);
//restrictToMatchingDesignations is a Grid Service Call

//Create a SetResolutionPolicy to handle the details of Resolving the CodedNodeSet
//Here, we will set the Maximum number of Concepts returned to 10.
SetResolutionPolicy resolvePolicy = new SetResolutionPolicy();
resolvePolicy.setMaximumToReturn(10);

//Do the resolve
ResolvedConceptReferenceList rcrlist = cnsg.resolveToList(resolvePolicy); //resolveToList is a Grid Service Call

//Use the returned ResolvedConceptReferenceList to print some details about the concepts found
ResolvedConceptReference[] rcref = rcrlist.getResolvedConceptReference();
for (int i = 0; i < rcref.length; i++) {
        System.out.println(rcref[i].getConceptCode());
        System.out.println(rcref[i].getReferencedEntry().
                getPresentation()[0].getText().getContent());
}
```

Error Handling

Error Connecting to LexEVS Grid Service

When connecting through the Java Client, java.net.ConnectException and org.apache.axis.types.URI.MalformedURIException may be thrown upon an unsuccessful attempt to connect.

A MalformedURIException is thrown in the case if a poorly-formed URL string. In this case, the exception is thrown before an attempt to connect is even made.

If the URL is well-formed, proper connection is tested. If the connection attempt fails, a ConnectException is thrown containing the reason for the failure.

```
try{
LexBIGServiceGridAdapter lbsg = new LexBIGServiceGridAdapter("http://localhost:8080/wsrf/services/cagrid
/LexEVSGridService");
} catch(java.net.ConnectException e){
        //Error Connecting
        e.printStackTrace();
} catch(org.apache.axis.types.URI.MalformedURIException e){
        //URL Syntax Error
        e.printStackTrace();
}
```

This example shows a typical connection to the LexEVS Grid Service, with the two potential Exceptions being caught and handled as necessary.

LexBIG Errors

LexBIG errors will be forwarded through the Distributed LexBIG layer and then on to the Grid layer. Input parameters, along with any other LexBIG (or Distributed LexBIG) errors will be detected on the server, not the client, and forwarded. All Generic LexBIG (or Distributed LexBIG) errors will be forwarded via a RemoteException, with the cause of the error and underlying LexBIG error message included.

Invalid Service Context Access

Service Context Services are not meant to be called directly. If the client attempts to do so, an `org.LexGrid.LexBIG.cagrid.LexEVSGridService.CodedNodeSet.stubs.types.InvalidServiceContextAccess` Exception will be thrown. This indicates a call was made to a Service Context without obtaining a Service Context Reference via the Main Service (see the above section Service Contexts and State for more information).

Security Issues

LexEVS Grid Service Security

Certain vocabulary content accessible through the LexEVS Grid Service may require extra authorization to access. Each client is required to supply its own access credentials via Security Tokens. These Security Tokens are implemented by a SecurityToken object:

Name: SecurityToken Namespace: gme://caCORE.caCORE/3.2/gov.nih.nci.evs.security Package: gov.nih.nci.evs.security

Accessing Secure Content

A client establishes access to a secured vocabulary via the following Grid Service Calls:

- *Step 1*: Connect to the LexBIG caGrid Service `LexBIGServiceGrid lbs = new LexBIGServiceGridAdapter(url);`
- *Step 2*: Build an `org.LexGrid.LexBIG.DataModel.cagrid.CodingSchemeIdentification` to hold the Coding Scheme name. `CodingSchemeIdentification codingScheme = new CodingSchemeIdentification(); codingScheme.setName ("codingScheme");`
- *Step 3*: Build an `gov.nih.nci.evs.security.SecurityToken` containing the security information for the desired Coding Scheme. `SecurityToken token = new SecurityToken (); token.setAccessToken("securityToken");`
- *Step 4*: Invoke the LexBIG caGrid service as follows: This will return a reference to a new "LexBIGServiceGrid" instance that is associated with the security properties that were passed in.
  `LexBIGServiceGrid lbsg = lbs.setSecurityToken(codingScheme, token);`

> **Note**
>
> The Grid Service "setSecurityToken" returns an `org.LexGrid.LexBIG.cagrid.LexEVSGridService.stubs.types . LexEVSGridServiceReference.LexEVSGridServiceReference` object. This reference must be used to access the secured vocabularies.

Implementation

Each call to "setSecurityToken" sets up a secured connection to Distributed LexBIG with the access privileges included in the SecurityToken parameter. The LexEVSGridServiceReference that is returned to the client contains a unique key identifier to the secure connection that has been created on the server. All subsequent calls the client makes through this LexEVSGridServiceReference will be made securely. If additional SecurityTokens are passed in through the "setSecurityToken" Grid Service, the additional security will be added and maintained.

The "setSecurityToken" Grid Service is a stateful service. This means that after the client sets a SecurityToken, any subsequent call will be applied to that SecurityToken.

Secure connections are not maintained on the server indefinitely, but are based on load conditions. The server will allow 30 unique secure connections to be set up for clients without any time limitations. As additional requests for secure connections are received by the server, connections will be released by the server on an 'oldest first' basis. No connection, however, may be released prior to 5 minutes after its creation.

If no SecurityTokens are passed in by the client, a non-secure Distributed LexBIG connection will be used. The server maintains one (and only one) un-secured Distributed LexBIG connection that is shared by any client not requesting security.

> **Note**
>
> All non-secured information accessed by the LexEVS Grid Service is publicly available from NCICB and users are expected to follow the licensing requirements currently in place for accessing and using NCI EVS information.

Related Links

- caGrid web page
- LexBIG Core Services