LexEVS 4.2 Grid Service Design and Implementation

⚠ The information and links on this page are no longer being updated and are provided for reference purposes only.

Unable to render {include} The included page could not be found.

Contents of this Page

Unable to render {include} The included page could not be found.

Revision History

Content changes to this document from the previous to the current level are indicated by revision bars (|) unless a complete rewrite is indicated.

Date	Version	Description	Author
07/29/2008	1.0	Initial document	Kevin Peterson
8/30/2008	1.1	Revised for Security and Exception Handling	Kevin Peterson

Note

If this document has been inspected, please indicate the inspection date that each version is based on in the "Change Description and Explanation" area. Entries in this log must be maintained for at least 3 years.

Document Purpose

This document provides the detailed design and implementation of LexBIG Enterprise Vocabulary Service (LexEVS) caGrid Service. It should be noted that the LexEVS Grid Service is no longer part of the caGrid 1.1 infrastructure and will be deployed as a separate unit. This is a change from the previous release of the LexEVS Grid Service.

The LexEVS caGrid service will allow programs to utilize the caGrid 1.2 infrastructure to access LexEVS information that is currently being produced by NCICB.

Implementation Overview

Team Members

The following table lists team members.

Role	Name
Development Lead	Kevin Peterson
Documentation Lead	Kevin Peterson
Project Manager	Tom Johnson

Description

The LexEVS grid service will be used to obtain data accessible via the EVS API 4.2 service, specifically, the Distributed LexBIG services. Please refer to the caCORE EVS API 4.0 technical guide for details on the EVS API 4.2 EVS APIs, and the LexBIG 2.3 Technical and Admin guide for details about LexBIG 2.3.

For more Documentation, Build/Deployment instructions and examples, refer to the project documentation on the GForge archive page.

Scope

The LexEVS Grid service will provide programmatic access to the LexBIG domain objects that are available via the LexBIG information model.

The LexEVS grid service will be registered in Cancer Data Standards Repository (caDSR) under the following category:

Category	LexEVS Grid Service
Context	caBIG
Classification Scheme	LexBIG
Version	LexBIG_v2_3_rv1

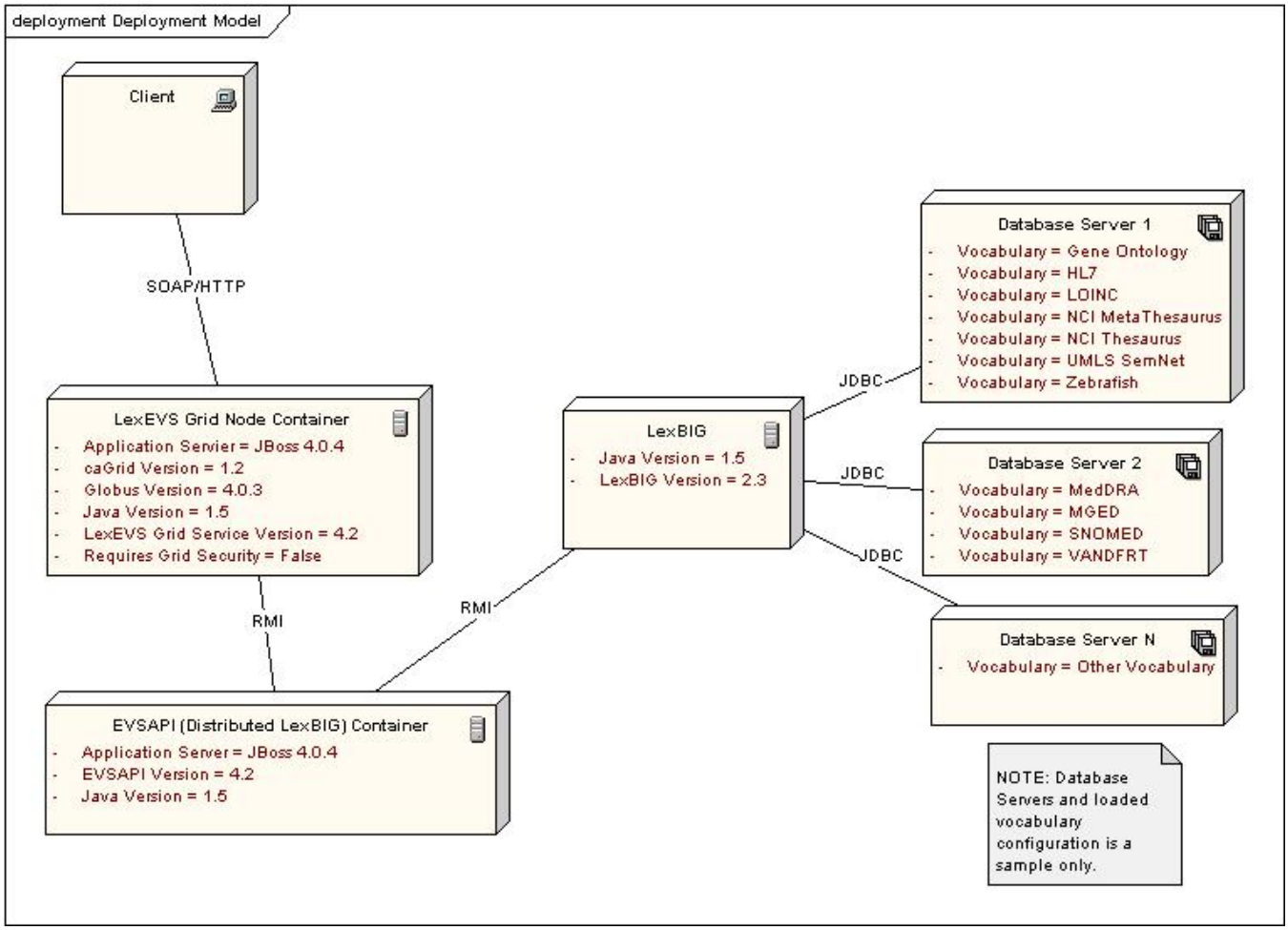
Architecture

The LexEVS Grid Service is implemented to expose the API and Model of LexBIG 2.3. For more information on LexBIG, see the Mayo Clinic website .

LexEVS Grid Service is deployed in a JBoss Application Server, inside of a Globus Web Application installation. LexEVS Grid Service depends on EVSAPI, which is also deployed to a JBoss container. For more information on the deployment of EVSAPI, see the GForge archive.

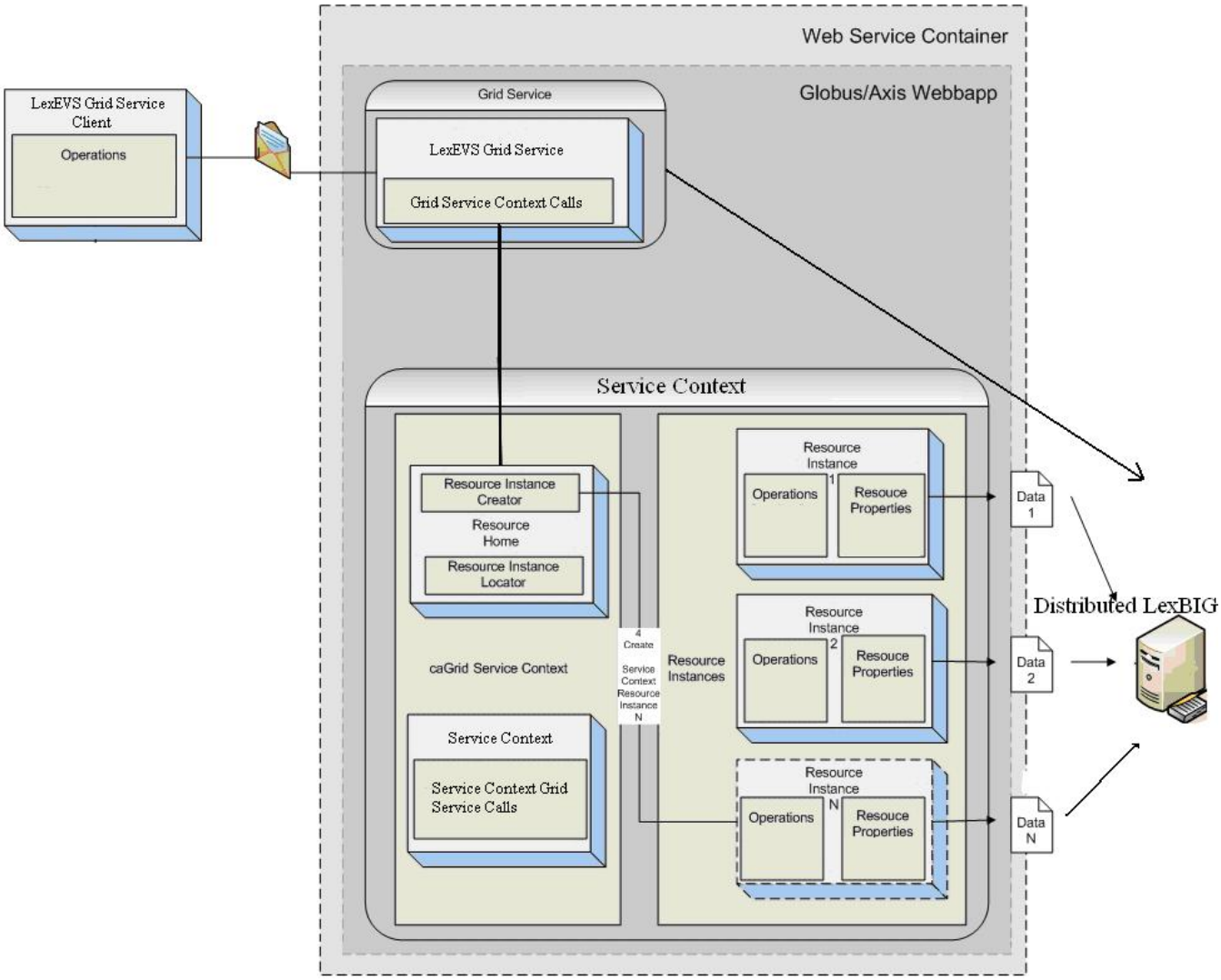
EVSAPI itself depends on an installation of LexBIG .

The diagram below shows the various components of the LexEVS Grid Service System and how they interact.



LexEVS Grid Service and EVSAPI need not be deployed to physically separate servers, but it is recommended that if they are co-located on the same server, they should be deployed to separate JBoss containers.

Below is the LexEVS Grid Service Architecture, viewed from inside of the Web Service Container. For more information on how Service Contexts and Resources are used, see the section, Service Contexts and State below.



LexEVS Grid Service Class Diagram

The LexEVS Grid Service is built on the LexGrid/LexBIG model and implementation. For more information about this model, visit:

```
Historical link
https://gforge.nci.nih.gov/plugins/scmsvn/viewcvs.php/LexBIG_Core_Services/LexBIG-2.3/lexbig/lbModel/?
root=lexevs
```

and

```
Historical link
https://gforge.nci.nih.gov/plugins/scmsvn/viewcvs.php/LexBIG_Core_Services/LexBIG-2.3/lgModel/?root=lexevs
```

Also, visit the Mayo website for background information as well as Class Diagrams, examples, and other information.

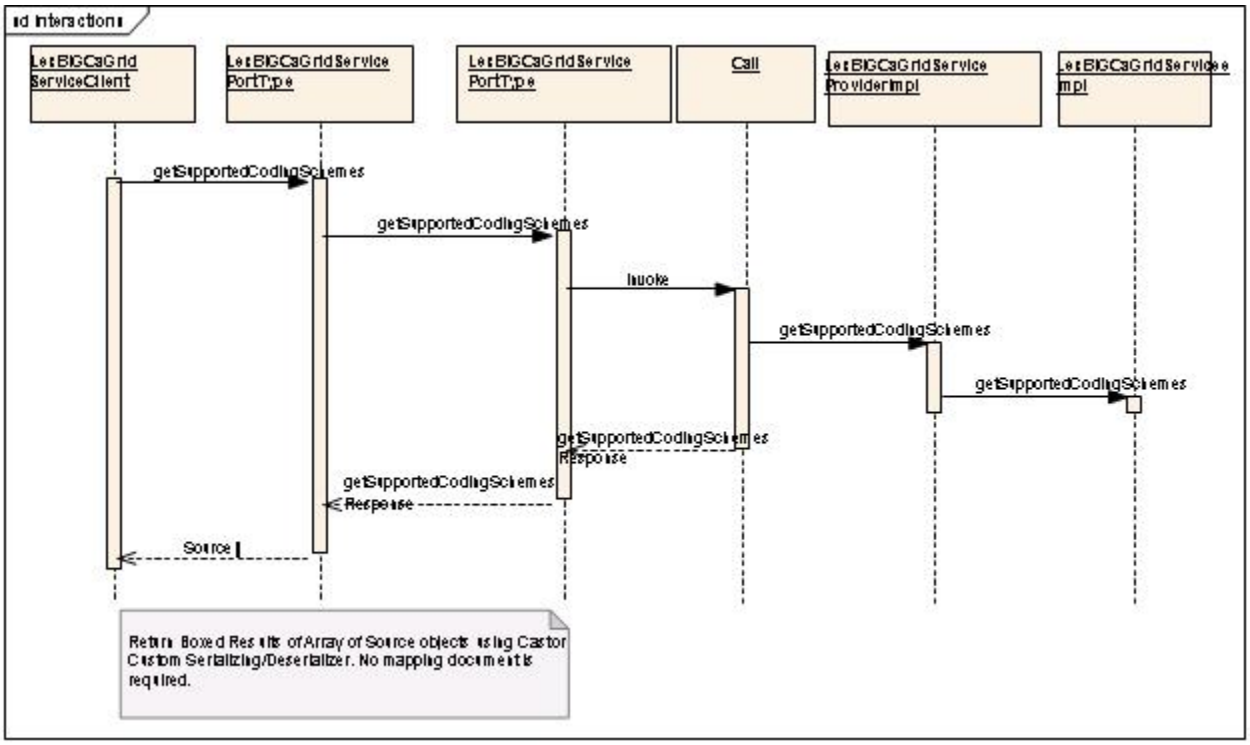
For information specific to the LexEVS Grid Service, visit:

```
Historical link
https://gforge.nci.nih.gov/plugins/scmsvn/viewcvs.php/LexBIG_Core_Services/LexBIG-2.3/lexbig/lbModel.cagrid/?
root=lexevs
```

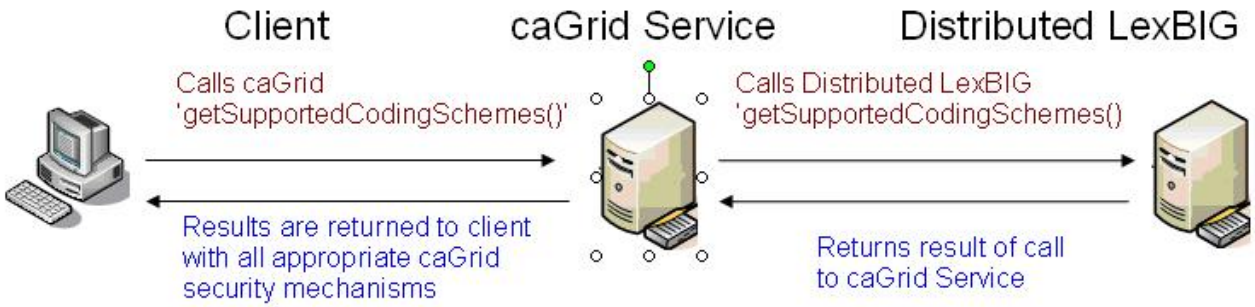
This link contains Class Diagrams and descriptions for input/output parameters, as well as other information concerning the Silver Level Compliance submission package.

LexEVS Grid Service Sequence Diagram

The sequence diagram for the operation "getSupportedCodingSchemes" is described below:



The following diagram shows a General Call Sequence from the client, through the caGrid Service, to the Distributed LexBIG and the returned results:



Assumptions

- The LexEVS service will be based on the latest EVSAPI 4.2 patch release built by NCICB.
- The LexBIG 2.3 domain model will be loaded in the GME and caDSR.
- The LexEVS Grid Service will not have any method level security. All security requirements will be handled by the actual deployment of the underlying EVSAPI 4.2 service. Please see the "Security" section below for more information on how the LexEVS Grid Service utilizes this security.
- The LexEVS Grid Service will not be deployed as a "core" service by caGrid at NCICB as was previously done, but rather will now be deployed as a standalone service.
- The LexEVS Grid Service release schedule will no longer be coupled to the caGrid deployment schedule as previously done.
- Multiple version of LexEVS Grid Service may be active at the same instance in time depending solely on the availability of the underlining EVSAPI service.

Dependencies

- EVSAPI 4.2 service needs to be available and running correctly.
- The LexBIG 2.3 domain model needs to be registered in caDSR.
- The LexEVS service and operations will use the Introduce toolkit to generate the appropriate structure for registering the service into caDSR.

Issues

- None

Third Party Tools

- Introduce Toolkit
- Globus Toolkit (4.0.3) or appropriate version supported by caGrid 1.2
- caGrid 1.2 core infrastructure

Implementation Contents

Server

The LexEVS Grid Service will be deployed as a "stand alone" grid service at NCICB.

Algorithms

None

Batch Processes

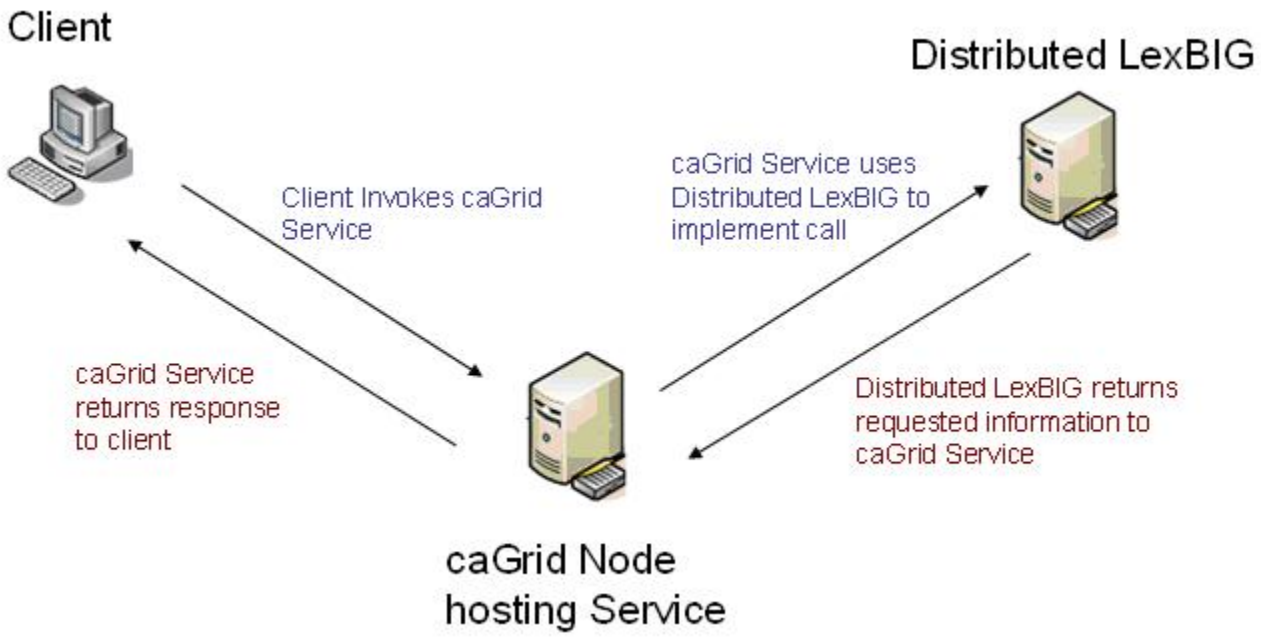
None

APIs

The main Service API exposed by the LexEVS Grid service will be the http://informatics.mayo.edu/LexGrid/downloads/javadocGrid/org/LexGrid/LexBIG/cagrid/interfaces/LexBIGServiceGrid.html Interface. All other APIs will not be directly exposed, but will be made available through Service Contexts.

In General, API calls will follow the sequence which is shown in the following diagram:

1. Client invokes caGrid service .
2. caGrid Service uses Distributed LexBIG to implement call.
3. Distributed LexBIG returns requested info to caGrid service.
4. caGrid service sends response to client.



Main Service API

http://informatics.mayo.edu/LexGrid/downloads/javadocGrid/org/LexGrid/LexBIG/cagrid/interfaces/LexBIGServiceGrid.html

Using the API

To use the LexEVS Grid Services, either org.LexGrid.LexBIG.cagrid.adapters.LexBIGServiceAdapter or org.LexGrid.LexBIG.cagrid.adapters.LexBIGServiceGridAdapter objects may be instantiated. These are two different Interfaces for accessing the Grid Services.

- **org.LexGrid.LexBIG.cagrid.adapters.LexBIGServiceAdapter**- 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 using the LexBIG API to used 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://...");
```

- **org.LexGrid.LexBIG.cagrid.adapters.LexBIGServiceGridAdapter**- An Interface for interacting with the LexEVS Grid Services. 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://...");
```

getCodingSchemeConcepts

getCodingSchemeConcepts(String, 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	1. Create a Resource on the server and populate it with the requested org.LexGrid.LexBIG.LexBIGService.CodedNodeSet. 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:	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);`
	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");`
	3. Build an org.LexGrid.LexBIG.DataModel.cagrid.CodingSchemeIdentification to hold the Coding Scheme name.
	`CodingSchemeIdentification codingScheme = new CodingSchemeIdentification();`
	`codingScheme.setCode(code);`
	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	1. Create a Resource on the server and populate it with the requested org.LexGrid.LexBIG.Extensions.Query.Filter 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	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);`
	2. Build an org.LexGrid.LexBIG.DataModel.cagrid.ExtensionIdentification to hold the Extension name.
	`ExtensionIdentification extension = new ExtensionIdentification();`
	`extension.setLexBIGExtensionName(name);`
	3. Invoke the LexBIG caGrid service as follows:
	`Filter filter = lbs.getFilter(extension);`

getSortAlgorithm

getSortAlgorithm(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.Sort.* *stubs.types.SortReference*
Exception	*RemoteException*
Implementation Details	1. Create a Resource on the server and populate it with the requested org.LexGrid.LexBIG.Extensions.Query.Sort 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	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);`
	2. Build an org.LexGrid.LexBIG.DataModel.cagrid.ExtensionIdentification to hold the Extension name.
	`ExtensionIdentification extension = new ExtensionIdentification();`
	`extension.setLexBIGExtensionName(name);`
	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	Call this method on the associated LexBIG Service instance (or Distributed LexBIG instance) on the server, and forward the results.

Sample Call	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);`
	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	1. Create a Resource on the server and populate it with the requested org.LexGrid.LexBIG.LexBIGService.LexBIGServiceMetadata 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	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);`
	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	Call this method on the associated LexBIG Service instance (or Distributed LexBIG instance) on the server, and forward the results.

Sample Call	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);`
	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	Call this method on the associated LexBIG Service instance (or Distributed LexBIG instance) on the server, and forward the results.

Sample Call:	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);`
	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	Call this method on the associated LexBIG Service instance (or Distributed LexBIG instance) on the server, and forward the results.

Sample Call	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);`
	2. Build an org.LexGrid.LexBIG.DataModel.cagrid.CodingSchemeIdentification to hold the Coding Scheme name.
	`CodingSchemeIdentification codingScheme = new CodingSchemeIdentification();`
	`codingScheme.setCode(code);`
	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");`
	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	1. Create a Resource on the server and populate it with the requested org.LexGrid.LexBIG.LexBIGService.CodedNodeGraph. 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	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);`
	2. Build an org.LexGrid.LexBIG.DataModel.cagrid.CodingSchemeIdentification to hold the Coding Scheme name.
	`CodingSchemeIdentification codingScheme = new CodingSchemeIdentification(); codingScheme.setCode(code);`
	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");`
	4. Build an org.LexGrid.LexBIG.DataModel.cagrid.RelationContainerIdentification containing the Relation Container information.
	`RelationContainerIdentification container = new RelationContainerIdentification(); container.setDc(name);`
	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:	Call this method on the associated LexBIG Service instance (or Distributed LexBIG instance) on the server, and forward the results.

Sample Call:	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);`
	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 that 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:	Call this method on the associated LexBIG Service instance (or Distributed LexBIG instance) on the server, and forward the results.

Sample Call:	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);`
	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:	Call this method on the associated LexBIG Service instance (or Distributed LexBIG instance) on the server, and forward the results.

Sample Call:	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.
	2. Build an org.LexGrid.LexBIG.DataModel.cagrid.ExtensionIdentification to hold the Extension name.
	`ExtensionIdentification extension = new ExtensionIdentification();`
	`extension.setLexBIGExtensionName("LexBIGServiceConvenienceMethods");`
	3. Invoke the LexBIG caGrid service as follows:
	LexBIGServiceConvenienceMethodsGrid `lbscm = lbs.getGenericExtensions(extension);`
	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:	1. Call this method on the associated LexBIG Service instance (or Distributed LexBIG instance) on the server, and forward the results. 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:	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);`
	2. Build an org.LexGrid.LexBIG.DataModel.cagrid.CodingSchemeIdentification to hold the Coding Scheme name.
	`CodingSchemeIdentification codingScheme = new CodingSchemeIdentification();`
	`codingScheme.setCode(code);`
	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:	Call this method on the associated LexBIG Service instance (or Distributed LexBIG instance) on the server, and forward the results.
Sample Call:	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);` 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:	Call this method on the associated LexBIG Service instance (or Distributed LexBIG instance) on the server, and forward the results.

Sample Call:	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);`
	2. Build an org.LexGrid.LexBIG.DataModel.cagrid.CodingSchemeIdentification to hold the Coding Scheme name.
	`CodingSchemeIdentification codingScheme = new CodingSchemeIdentification();`
	`codingScheme.setCode(code);`
	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");`
	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:	Call this method on the associated LexBIG Service instance (or Distributed LexBIG instance) on the server, and forward the results.

Sample Call:	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);`
	2. Build an org.LexGrid.LexBIG.DataModel.cagrid.CodingSchemeIdentification to hold the Coding Scheme name.
	`CodingSchemeIdentification codingScheme = new CodingSchemeIdentification();`
	`codingScheme.setName("codingScheme");`
	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");`
	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);`

API Examples

For an example clients, service calls, and SOAP messages, see the sample code on the LexEVS documentation GForge archive page.

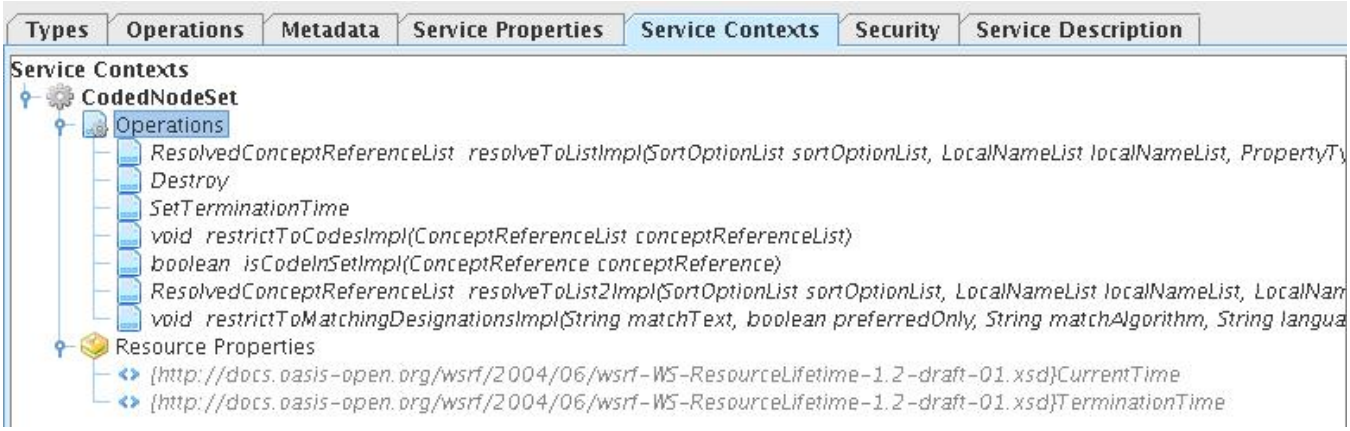
Example API usage:

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());
}
```

Service Contexts and State

Along with the Main Service (described above), the Server will also host the Service Contexts shown in the following diagram and described in the subsequent sections. These Service Contexts are not meant to be called directly as Grid Services. The main function of these Service Contexts is to provide additional functionality to the Main Service.

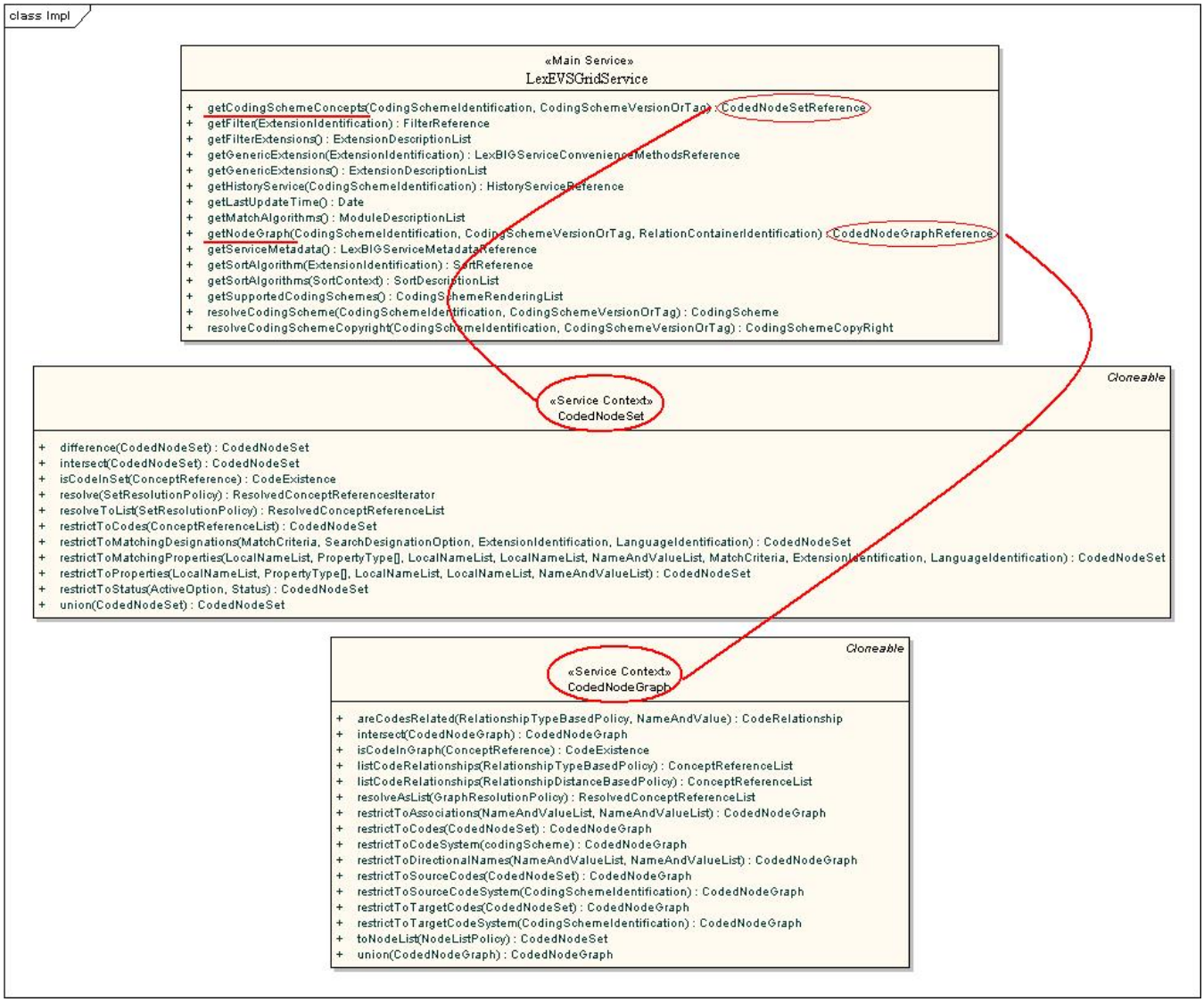


Service Context Operations Example in Introduce

> **Important**
>
> Service Contexts are only meant to be called through the Main Service - not directly. Through the Main Service, References to these Service Contexts can be obtained. Calls are made to the Service Contexts through these References.

Obtaining a Service Context Reference

In the figure below, two LexEVS Grid Service Calls are highlighted, 'getCodingSchemeConcepts' and 'getNodeGraph'. These two Grid Service Calls have been selected because they return to the user a "Reference" to a Service Context. For 'getCodingSchemeConcepts', the return type is CodedNodeSetReference (which references the CodedNodeSet Service Context). For 'getNodeGraph', the return type is CodedNodeGraphReference (which references the CodedNodeGraph Service Context).



Resources

LexEVS Grid Services use the WS-Resource Framework (WSRF) to allow for stateful calls to the server. When a client requests a Service Context, the client is not only issued a Reference to the Service Context that was requested, but to a unique stateful Resource on the server as well. This Resource is used in the LexEVS Grid Services as a way of statefully holding objects for further use by the client. For more information about how caGrid uses the WS-Resource Framework (WSRF), see http://www.cagrid.org/wiki/Metadata:WSRF

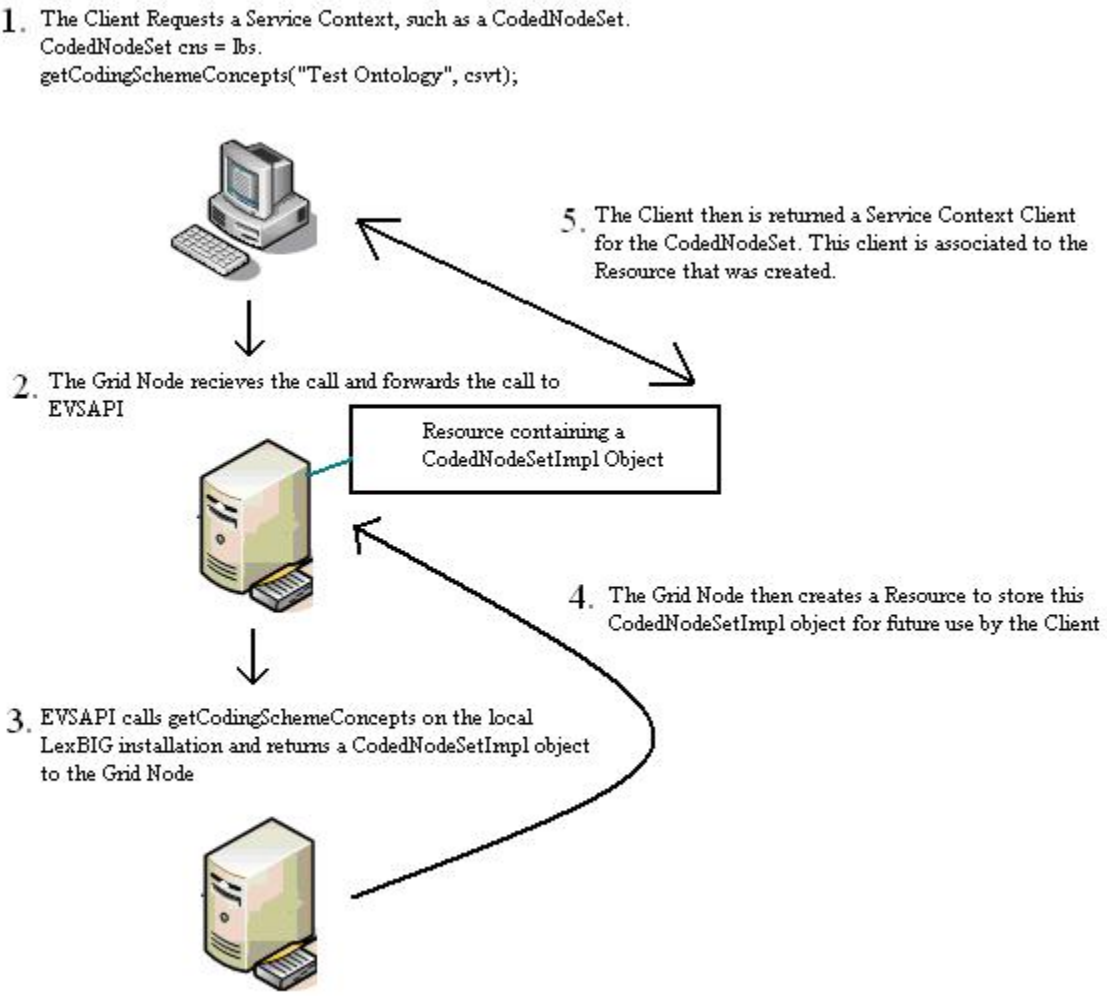
```
Historical link
http://www.cagrid.org/wiki/Metadata:WSRF
```

For more information on how Resources are implemented in the LexEVS Grid Service, refer to this LexEVSGrid presentation.

Service Context Sequence

The Service Contexts API calls follow this general process:

1. Client Requests a Service Context, such as a CodedNodeSet.
2. The Grid Node receives the call and forwards the call to EVSAPI.
3. EVSAPI calls getCodingSchemeConcepts on the local LexBIG installation and returns a CodedNodeSetImpl object to the Grid Node.
4. The Grid Node then creates a Resource to store this CodedNodeSetImpl object for future use by the Client.
5. The Client is then returned a Service Context Client for the CodedNodeSet. This client is associated to the Resource that was created.



Service Context and Resource Assignment

> **Note**
>
> By default, these services are destroyed 5 minutes after creation.

Supported Service Contexts.

CodedNodeSet

http://informatics.mayo.edu/LexGrid/downloads/javadocGrid/org/LexGrid/LexBIG/cagrid/interfaces/CodedNodeSetGrid.html

To construct a CodedNodeSet, the user calls getCodingSchemeConcepts as described above. When the user creates a CodedNodeSet through the API call getCodingSchemeConcepts, the server creates and stores the CodedNodeSet server-side as a Resource. This Resource is associated with the client and will be accessible only by the client that created it.

CodedNodeSet Call Sequence:

1. The user requests a CodedNodeSet using getCodingSchemeConcepts.

Java Code Snippet

```
LexBIGService lbs  = (LexBIGService)ApplicationServiceProvider.getApplicationServiceFromUrl(serviceUrl,
"EvsServiceInfo");
CodedNodeSet cns = lbs.getCodingSchemeConcepts(
        String codingScheme,
        org.LexGrid.LexBIG.DataModel.Core.CodingSchemeVersionOrTag);
```

2. The server calls the Distributed LexBIG getCodingSchemeConcepts method, returning to the server an org.LexGrid.LexBIG.Impl.CodedNodeSetImpl (the implementation of org.LexGrid.LexBIG.LexBIGService.CodedNodeSet) object.
3. The server then creates an org.LexGrid.LexBIG.cagrid.LexBIGCaGridServices.CodedNodeSet.service.globus.resource.CodedNodeSetResource. This Resource will be used to hold the instance of org.LexGrid.LexBIG.Impl.CodedNodeSetImpl, the implementation of org.LexGrid.LexBIG.LexBIGService.CodedNodeSet that was created above.
4. The server returns an org.LexGrid.LexBIG.cagrid.LexBIGCaGridServices.CodedNodeSet.stubs.types.CodedNodeSetReference object to the client. This is the reference to the CodedNodeSet Service Context. This object has a direct reference to the Resource created above. The user now uses this client to make transparent Grid calls through the Service Context.
5. The client may continue to make statefull calls to the CodedNodeSetClient and the assigned Resource.
6. These restrictions are separate calls but statefully maintained on the server via the Resource.

CodedNodeGraph

http://informatics.mayo.edu/LexGrid/downloads/javadocGrid/org/LexGrid/LexBIG/cagrid/interfaces/CodedNodeGraphGrid.html

To construct a CodedNodeGraph, the user calls getNodeGraph as described above. When the user creates a CodedNodeGraph through the API call getNodeGraph, the server creates and stores the CodedNodeGraph server-side as a Resource. This Resource is associated with the client and will be accessible only by the client that created it.

CodedNodeGraph Call Sequence:

1. The user requests a CodedNodeGraph using getCodingSchemeConcepts.

Java Code Snippet

```
LexBIGService lbs  = (LexBIGService)ApplicationServiceProvider.getApplicationServiceFromUrl(serviceUrl,
"EvsServiceInfo");
CodedNodeGraph cng = lbs.getNodeGraph(
                                  String codingScheme,
                                  org.LexGrid.LexBIG.DataModel.Core.CodingSchemeVersionOrTag,
                                  String relationsContainerName (Optional);
```

2. The server calls the Distributed LexBIG getNodeGraph method, returning to the server an org.LexGrid.LexBIG.Impl.CodedNodeGraphImpl (the implementation of org.LexGrid.LexBIG.LexBIGService.CodedNodeGraph) object.
3. The server then creates an org.LexGrid.LexBIG.cagrid.LexBIGCaGridServices.CodedNodeGraph.service.globus.resource.CodedNodeGraphResource. This Resource will be used to hold the instance of org.LexGrid.LexBIG.Impl.CodedNodeGraphImpl, the implementation of org.LexGrid.LexBIG.LexBIGService.CodedNodeGraph that was created above.
4. The server returns an org.LexGrid.LexBIG.cagrid.LexBIGCaGridServices.CodedNodeGraph.stubs.types.CodedNodeGraphReference object to the client. This is the reference to the CodedNodeGraph Service Context. This object has a direct reference to the Resource created above. The user now uses this client to make transparent Grid calls through the Service Context.
5. The client may continue to make statefull calls to the CodedNodeGraphClient and the assigned Resource. For example, the client may add Restrictions to the CodedNodeGraph before a Resolve:

Java Code Snippet

```
    cng.restrictToCodeSystem(org.LexGrid.LexBIG.DataModel.cagrid.CodingSchemeIdentification);
```

6. These restrictions are separate calls but statefully maintained on the server via the Resource.

LexBIGServiceConvenienceMethods

http://informatics.mayo.edu/LexGrid/downloads/javadocGrid/org/LexGrid/LexBIG/cagrid/interfaces/LexBIGServiceConvenienceMethodsGrid.html

To construct a LexBIGServiceConvenienceMethods, the user calls getGenericExtensions as described above. When the user creates a LexBIGServiceConvenienceMethods through the API call getGenericExtensions, the server creates and stores the LexBIGServiceConvenienceMethods server-side as a Resource. This Resource is associated with the client and will be accessible only by the client that created it.

LexBIGServiceConvenienceMethods Call Sequence:

1. The user requests a LexBIGServiceConvenienceMethods using getGenericExtensions.

Java Code Snippet

```
    LexBIGServiceConvenienceMethodsGrid lbscm = lbs.getGenericExtensions(org.LexGrid.LexBIG.DataModel.
cagrid.ExtensionIdentification);
```

2. The server calls the Distributed LexBIG getGenericExtensions method, returning to the server an org.LexGrid.LexBIG.Impl.Extensions.GenericExtensions.LexBIGServiceConvenienceMethodsImpl (the implementation of org.LexGrid.LexBIG.Extensions.Generic.LexBIGServiceConvenienceMethods) object.
3. The server then creates an org.LexGrid.LexBIG.cagrid.LexBIGCaGridServices.LexBIGServiceConvenienceMethods.service.globus.resource.LexBIGServiceConvenienceMethodsResource. This Resource will be used to hold the instance of org.LexGrid.LexBIG.Impl.Extensions.GenericExtensions.LexBIGServiceConvenienceMethodsImpl, the implementation of org.LexGrid.LexBIG.Extensions.Generic.LexBIGServiceConvenienceMethods that was created above.
4. The server returns an org.LexGrid.LexBIG.cagrid.LexBIGCaGridServicesLexBIGServiceConvenienceMethods.stubs.types.LexBIGServiceConvenienceMethodsReference object to the client. This is the reference to the LexBIGServiceConvenienceMethods Service Context. This object has a direct reference to the Resource created above. This LexBIGServiceConvenienceMethodsClient implements org.LexGrid.LexBIG.Extensions.Generic.LexBIGServiceConvenienceMethods. The user now uses this client to make transparent Grid calls through the Service Context. Because this LexBIGServiceConvenienceMethods implements org.LexGrid.LexBIG.Extensions.Generic.LexBIGServiceConvenienceMethods, API calls will look to the user as being identical to direct LexBIG API calls.
5. The client may continue to make statefull calls to the LexBIGServiceConvenienceMethods Client and the assigned Resource.
6. These API calls are separate calls but statefully maintained on the server via the Resource.

LexBIGServiceMetadata

http://informatics.mayo.edu/LexGrid/downloads/javadocGrid/org/LexGrid/LexBIG/cagrid/interfaces/LexBIGServiceMetadataGrid.html

To construct a LexBIGServiceMetadata, the user calls getServiceMetadata as described above. When the user creates a LexBIGServiceMetadata through the API call getServiceMetadata , the server creates and stores the LexBIGServiceMetadata server-side as a Resource. This Resource is associated with the client and will be accessible only by the client that created it.

LexBIGServiceMetadata Call Sequence:

1. The user requests a LexBIGServiceMetadata using getServiceMetadata.

Java Code Snippet

```
    LexBIGServiceMetadataGrid metadata = lbs.getServiceMetadata();
```

2. The server calls the Distributed LexBIG getServiceMetadata method, returning to the server an implementation of org.LexGrid.LexBIG.LexBIGService.LexBIGServiceMetadata object.
3. The server then creates an org.LexGrid.LexBIG.cagrid.LexBIGCaGridServices.LexBIGServiceMetadata.service.globus.resource.LexBIGServiceMetadataResource. This Resource will be used to hold the instance of an implementation of org.LexGrid.LexBIG.LexBIGService.LexBIGServiceMetadata.
4. org.LexGrid.LexBIG.cagrid.LexBIGCaGridServices.LexBIGServiceMetadata.stubs.types.LexBIGServiceMetadata object to the client. This is the reference to the LexBIGServiceMetadata Service Context. This object has a direct reference to the Resource created above. The user now uses this client to make transparent Grid calls through the Service Context.
5. The client may continue to make statefull calls to the LexBIGServiceMetadata and the assigned Resource.
6. These API calls are separate calls but statefully maintained on the server via the Resource.

HistoryService

http://informatics.mayo.edu/LexGrid/downloads/javadocGrid/org/LexGrid/LexBIG/cagrid/interfaces/HistoryServiceGrid.html

To construct a HistoryService, the user calls getHistoryService as described above. When the user creates a HistoryService through the API call getHistoryService, the server creates and stores the HistoryService server-side as a Resource. This Resource is associated with the client and will be accessible only by the client that created it.

HistoryService Call Sequence:

1. The user requests a HistoryService using getHistoryService .

Java Code Snippet

```
    HistoryServiceGrid history = lbs.getHistoryService(org.LexGrid.LexBIG.DataModel.cagrid.
CodingSchemeIdentification);
```

2. The server calls the Distributed LexBIG getHistoryService method, returning to the server an implementation of org.LexGrid.LexBIG.History.HistoryService object.

3. The server then creates an org.LexGrid.LexBIG.cagrid.LexBIGCaGridServices.HistoryService.service.globus.resource.HistoryServiceResource. This Resource will be used to hold the instance of an implementation of org.LexGrid.LexBIG.History.HistoryService.
4. The server returns an org.LexGrid.LexBIG.cagrid.LexBIGCaGridServices.LexBIGServiceMetadata.stubs.types.LexBIGServiceMetadata object to the client. This is the reference to the HistoryService Service Context. This object has a direct reference to the Resource created above. The user now uses this client to make transparent Grid calls through the Service Context.
5. The client may continue to make statefull calls to the HistoryServiceClient and the assigned Resource. For example, the client may call any method in org.LexGrid.LexBIG.History.HistoryService. Example: history.getLatestBaseline();
6. These API calls are separate calls but statefully maintained on the server via the Resource.

Sort

http://informatics.mayo.edu/LexGrid/downloads/javadoc/org/LexGrid/LexBIG/Extensions/Query/Sort.html

To construct a Sort, the user calls getSortAlgorithm as described above. When the user creates a Sort through the API call getSortAlgorithm, the server creates and stores the Sort server-side as a Resource. This Resource is associated with the client and will be accessible only by the client that created it.

Sort Call Sequence:

1. The user requests a Sort using getSortAlgorithm .

Java Code Snippet

```
Sort sort = lbs.getSortAlgorithm(org.LexGrid.LexBIG.DataModel.cagrid.ExtensionIdentification);
```

2. The server calls the Distributed LexBIG getSortAlgorithm method, returning to the server an implementation of org.LexGrid.LexBIG.Extensions.Query.Sort) object.
3. The server then creates an org.LexGrid.LexBIG.cagrid.LexBIGCaGridServices.Sort .service.globus.resource.Sort Resource. This Resource will be used to hold the instance of an implementation of org.LexGrid.LexBIG.Extensions.Query.Sort.
4. The server returns an org.LexGrid.LexBIG.cagrid.LexBIGCaGridServices.service.SortClient object to the client. This is the client to the Sort Service Context. This object has a direct reference to the Resource created above. This SortClient implements org.LexGrid.LexBIG.Extensions.Query.Sort. The user now uses this client to make transparent Grid calls through the Service Context. Because this Sort implements org.LexGrid.LexBIG.Extensions.Query.Sort, API calls will look to the user as being identical to direct LexBIG API calls.
5. The client may continue to make statefull calls to the SortClient and the assigned Resource. For example, the client may call any method in org.LexGrid.LexBIG.Extensions.Query.Sort

Java Code Snippet

```
sort.compare(codedNodeReference1, codedNodeReference2);
```

6. These API calls are separate calls but statefully maintained on the server via the Resource.

Filter

http://informatics.mayo.edu/LexGrid/downloads/javadoc/org/LexGrid/LexBIG/Extensions/Query/Filter.htm

To construct a Filter, the user calls getFilter as described above. When the user creates a Filter through the API call getFilter, the server creates and stores the Sort server-side as a Resource. This Resource is associated with the client and will be accessible only by the client that created it.

Filter Call Sequence:

1. The user requests a Filter using getFilter

Java Code Snippet

```
Filter filter = lbs.getFilter(org.LexGrid.LexBIG.DataModel.cagrid.ExtensionIdentification);
```

2. The server calls the Distributed LexBIG getFilter method, returning to the server an implementation of org.LexGrid.LexBIG.Extensions.Query.Filter) object.
3. The server then creates an org.LexGrid.LexBIG.cagrid.LexBIGCaGridServices.Filter.service.globus.resource.FilterResource. This Resource will be used to hold the instance of an implementation of org.LexGrid.LexBIG.Extensions.Query.Filter.
4. The server returns an org.LexGrid.LexBIG.cagrid.LexBIGCaGridServices.service.FilterClient object to the client. This is the client to the Filter Service Context. This object has a direct reference to the Resource created above. This FilterClient implements org.LexGrid.LexBIG.Extensions.Query.Filter. The user now uses this client to make transparent Grid calls through the Service Context. Because this Filter implements org.LexGrid.LexBIG.Extensions.Query.Filter, API calls will look to the user as being identical to direct LexBIG API calls.
5. The client may continue to make statefull calls to the FilterClient and the assigned Resource. For example, the client may call any method in org.LexGrid.LexBIG.Extensions.Query.Filter

Java Code Snippet

```
    filter.match(resolvedConceptReference);
```

6. These API calls are separate calls but statefully maintained on the server via the Resource.

ResolvedConceptReferencesIterator

http://informatics.mayo.edu/LexGrid/downloads/javadoc/org/LexGrid/LexBIG/Utility/Iterators/ResolvedConceptReferencesIterator.html

A ResolvedConceptReferencesIterator is created when a CodedNodeSet or CodedNodeGraph is resolved. It allows results to be returned from the server incrementally instead of all at once. When the user creates a ResolvedConceptReferencesIterator, the server creates and stores the ResolvedConceptReferencesIterator server-side as a Resource. This Resource is associated with the client and will be accessible only by the client that created it.

ResolvedConceptReferencesIterator Call Sequence:

1. The user gets a ResolvedConceptReferencesIterator from a Resolve.
2. The server calls the Distributed LexBIG resolve method on the CodedNodeSet, returning to the server an implementation of org.LexGrid.LexBIG.Utility.Iterators.ResolvedConceptReferencesIterator object.
3. The server then creates an org.LexGrid.LexBIG.cagrid.LexBIGCaGridServices.ResolvedConceptReferencesIterator.service.globus.resource.ResolvedConceptReferencesIteratorResource. This Resource will be used to hold the instance of an implementation of org.LexGrid.LexBIG.Utility.Iterators.ResolvedConceptReferencesIterator.
4. The server returns an org.LexGrid.LexBIG.cagrid.LexBIGCaGridServices.service.ResolvedConceptReferencesIteratorClient object to the client. This is the client to the ResolvedConceptReferencesIterator Service Context. This object has a direct reference to the Resource created above. This ResolvedConceptReferencesIteratorClient implements org.LexGrid.LexBIG.Utility.Iterators.ResolvedConceptReferencesIterator. The user now uses this client to make transparent Grid calls through the Service Context. Because this ResolvedConceptReferencesIterator implements org.LexGrid.LexBIG.Utility.Iterators.ResolvedConceptReferencesIterator, API calls will look to the user as being identical to direct LexBIG API calls.
5. The client may continue to make statefull calls to the ResolvedConceptReferencesIteratorClient and the assigned Resource. For example, the client may call any method in org.LexGrid.LexBIG.Utility.Iterators.ResolvedConceptReferencesIterator

Java Code Snippet

```
    while(itr.hasNext){
         ResolvedConceptReference ref = itr.next();
    }
```

6. These API calls are separate calls but statefully maintained on the server via the Resource.

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.

Java Code

```
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).

Database Changes

None

Client

The Introduce toolkit generates a "client" class that will be provided to the users.

JSP/HTML

None

Servlet

None

Security Issues

Security in the LexEVS Grid Service is implemented in the Distributed LexBIG layer. The information in this section explains how the LexEVS Grid Services utilize this security implementation. For more information about the Distributed LexBIG Security Implementation, refer to LexBIG Access to Licensed Vocabulary Implementation, attached to the EVS API GForge documents archive.

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:

1. Connect to the LexBIG caGrid Service
   `LexBIGServiceGrid lbs = new LexBIGServiceGridAdapter(url);`
2. Build an org.LexGrid.LexBIG.DataModel.cagrid.CodingSchemeIdentification to hold the Coding Scheme name.

   ```
   CodingSchemeIdentification codingScheme = new CodingSchemeIdentification();
   codingScheme.setName("codingScheme");
   ```

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");
   ```

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);`

It is important to note that 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 of Security

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.

Performance

The LexEVS service will take advantage of all improvements made to the EVSAPI services with the exception of lazy loading. LexEVS grid service, being in nature a web service is currently not taking advantage of lazy loading since objects are transferred as fully populated objects. However, future releases of LexEVS Grid Service may refractor the interface in such as way as to take advantage of some of the benefits brought about by the inclusion of lazy loading in to EVSAPI service.

LexEVS Grid Services utilize the performance enhancements of the LexBIG API.
For more information about LexBIG performance (which LexEVS Grid Services are dependent on), see the Mayo Clinic website .

Internationalization

Not Internationalized

Installation / Packaging

The service will be installed and deployed as a "stand alone" service at NCICB.

Migration

Both the current version of LexEVS grid service and a previous version may be "in service" simultaneously if the corresponding underlying EVSAPI service is also "in service" to manage migration of clients.

System Testing

See LexEVS Grid Service Testing Documentation (LexEVS Grid Service System Testing in the Project Documents, Development Documents section)