LexEVS 6.5.4 Architecture

LexEVS GraphResolve Final Proposal

Final Proposal Description and Sign Off

Requirements:

1. Create a method over the Graph API that allows a CodedNodeSet to be passed as a parameter along with an Enumeration value indicating direction, a value for the depth of the resolution, and one or more associations. Return a List of Minimally populated ResolvedConceptReferences.
2. Constraints:
   - Will not require current RMI users to update their client jar
3. Features:
   - Will provide a loader for the graph database
   - Will provide instructions on the use of the loader, including any dependencies on existing services
4. Architecture:
   - Specify which Graph DB will need to be installed on which tiers
   - Specify any other services that need to be installed on which tiers

1. Create a Method:

Method Signature

public List<ResolvedConceptReference> getAssociatedConcepts(CodedNodeSet cns, Direction direction, int depth, NameAndValueList association);

Parameters:

Parameter cns: This CodedNodeSet must have a set of restrictions appropriate for query building and fully ready to be resolved. Cannot be null

Parameter direction: incoming or outgoing edges will be designated by one of these Enumerations. Cannot be null

Parameter depth: This allows depth control of the query including resolving only neighbors or a full resolution if depth is known. Entering -1 allows full resolution, 0 will return null

Parameter associations: The name or names of the edges in the graph (Must exist as a supported association the code system). Null returns all associations

Return Value List<ResolvedConceptReference>: a list of minimally populated concept references including code, namespace,

entity description and coding scheme uri and version. These objects are the result of a graph resolution without

any indication of where they existed in the graph before the resolution.

Method Behavior

This method requires some knowledge of building queries into the LexEVS system's CodedNodeSet API, including the capability

of building a CodedNodeSet set of restrictions through restriction method calls. Within the scope of this method,

the CodedNodeSet will be resolved to a ResolvedConceptReferenceList using the method

```
resolveToList(

SortOptionList sortOptions, LocalNameList propertyNames,

PropertyType[] propertyTypes, int maxToReturn)

throws LBInvocationException,LBParameterException;
```

The parameter set for this method will be defaulted to the following:

SortOptionList: null – No sort options allowed

LocalNameList: null – No restrictions on property names

PropertyType: null – No restrictions on property types

int: 10 Maximum return limited to ten entities.

Null value for associations will return values for all associations. Otherwise queries will be generated depending on each

association name.

Exceptions would be handled in this method and an appropriately messaged RuntimeException would be thrown on failure.

The ResolvedConceptReference objects returned contain only the code, name space, entityDescription, coding scheme URI, and coding scheme version.

It will not contain any entities or their properties or targetOf or sourceOf links to other entities.

2. Constraints

Initial testing is complete. Will retest before release. Since this is not implemented in the RMI application we don't need to require runtime client changes, especially since extensions are called as plugins and not from local client code.

3. Features

We have a loader complete at this point and it should have the same profile as other loaders.

The loader requires a local install of Arangodb along with a configuration of LexEVS where lbconfig.props is updated with connection parameters. Instructions for loader use will be supplied in the usual loader documentation section of the Wiki

Loader use is documented here:

Loading the Graph Database into ArangoDb for a Given Terminology

4. Architecture

We will work with the Systems team to get the most optimal installation of Arangodb for each tier.

Each tier will also need to have a tomcat instance with the graph-resolve installed to produce REST services for the Graph Resolution API. We'll provide the usual configuration parameters for the REST service that are normally defined for deployment tracks on Jenkins and/or PTE documents.

9 - Installing the Graph Node Resolution REST Service

5. Migration

LexEVS 6.5.4 Release Notes

Proposal Sign off

Federal Sponsor(s)	Signature
Lyubov Remennik	via email
Sherri De Coronado	SDC

User(s)	Signature
Kim Ong	via email

NCI System Architect	Signature
Tracy Safran	approved during arch meeting 2020.01.14

LexEVS Rest Service API

List of terminologies

base url/databases

List of associations

base url/graphDbs/<coding scheme name from database list>

Get inbound edges for coding scheme, association and unique id.

base url/getInbound/<coding scheme name>/<association name from graph list>/<entity code>

Get inbound edges for depth, coding scheme, association and unique id.

base url/getInbound/<depth/<coding scheme name>/<association name from graph list>/<entity code>

Get outbound edges for coding scheme, association and unique id.

base url/getOutbound/<coding scheme name>/<association name from graph list>/<entity code>

Get outbound edges for depth, coding scheme, association and unique id.

base url/getOutbound/<depth/<coding scheme name>/<association name from graph list>/<entity code>

LexEVS GraphResolve Legacy Requirements and Knowledge Base

Use Case/Requirements Statement:

The requirement is to provide an iterator that contains results of a relationship search:

(1) Given a coding scheme (name and version) and an association name, such as Anatomical_Structure_Is_a_Physical_Part_Of in NCI Thesaurus,

find all source concepts related to any target concept through this specified association such that the target concept matches with a collection of user specified search criteria.

(Refer to the NCI Thesaurus Advanced Search web page for user-specified search criteria)

(2) Given a coding scheme (name and version) and an association name, such as Anatomical_Structure_Is_a_Physical_Part_Of in NCI Thesaurus,

find all target concepts related to any source concept through this specified association such that the source concept matches with a collection of user specified search criteria.

(Refer to the NCI Thesaurus Advanced Search web page for user-specified search criteria)

(3) If no association name is specified, then search all supported associations.

Knowledge Base/FAQ

How are we performing searches in The NodeGraphResolutionExtensionImpl class

The search is being performed through the text searches available in the CodedNodeSet API and matched to the methods referencing NCI term browser regular and advanced search methods. This will be wrapped in a hybrid API where coded node entity codes are being referenced for a resolution against the graph database. The graph database can return enough information to construct basic ConceptReferences containing entity code and namespace attributes. The current method signature is for Iterator<ConceptReference> and iterates over the parent or child total graph resolution for one or more text search results. (I'm currently limiting this to 10). I am working on getting a query into the LexEVS database that provides a prescreening of any text search results that do not have a valid presence in the designated association. (either does not participate in the relationship or only participates where it's target or source is anonymous)

Is the graphing database a micro service?

We have implemented a spring boot micro service over the instance of Arangodb which is in turn loaded with graph edges from terminologies. This micro service has an JSON interface that would be available to users. This service currently only provides complete resolution of parents or children for a given vertex identifier (entity code). We could provide other options for this service, but this one allows the high performing return of values and therefore an accurate count where appropriate for a given iterator or list.

Does this include or support LexEVS Coded Node Graph APIs

No coded node graph API would be included in or supported by the micro service. It will serve up JSON result lists of code/namespace attributes only.

Can it be easily integrated with SPARQL?

We currently load the Arango graph database from LexEVS. There is no current or compelling reason we can't perform the same load from a triple store.

Project Timeline Record

Timeline for LexEVS Graph Service Design Discussion and Implementation.docx