Timeline for LexEVS Graph Service Design Discussion and Implementation

Original request from Kim including requirements, July 31st.

Earliest discussion in the Weekly meetings August 14th and 21st:

[https://wiki.nci.nih.gov/display/LexEVS/LexEVS+Meeting+Minutes+-+2019.08.21](https://wiki.nci.nih.gov/display/LexEVS/LexEVS%2BMeeting%2BMinutes%2B-%2B2019.08.21)

Earliest discussion in the Architecture group August 24th:

[https://wiki.nci.nih.gov/display/LexEVS/2019-08-27+Meeting+notes](https://wiki.nci.nih.gov/display/LexEVS/2019-08-27%2BMeeting%2Bnotes)

Earliest discussion in the Weekly meetings August 14th and 21st:

[https://wiki.nci.nih.gov/display/LexEVS/LexEVS+Meeting+Minutes+-+2019.08.21](https://wiki.nci.nih.gov/display/LexEVS/LexEVS%2BMeeting%2BMinutes%2B-%2B2019.08.21)

September 17th 1st hour long design discussion meeting

September 18th Weekly Technical Meeting we decide as a group to implement this and do so as a micro service. (Following the previous days design discussion.) Node and graph edge definitions were described.

[https://wiki.nci.nih.gov/display/LexEVS/LexEVS+Meeting+Minutes+-+2019.09.18](https://wiki.nci.nih.gov/display/LexEVS/LexEVS%2BMeeting%2BMinutes%2B-%2B2019.09.18)

September 24th Architecture meeting we reach consensus on a number of design and implementation points.

[https://wiki.nci.nih.gov/display/LexEVS/2019-09-24+Meeting+notes](https://wiki.nci.nih.gov/display/LexEVS/2019-09-24%2BMeeting%2Bnotes)

October 8th, Architecture meeting we discuss how the implementation might be used and how it has been tested

[https://wiki.nci.nih.gov/display/LexEVS/2019-10-08+Meeting+notes](https://wiki.nci.nih.gov/display/LexEVS/2019-10-08%2BMeeting%2Bnotes)

October 9th, one hour code review.

October 16th, one hour code review.

October 23rd, one hour long code review.

October 24th First commit of an API to the code base:

<https://github.com/lexevs/lexevs/commit/3ef0f4bccd3bf6504a689c8207bc61a12a85106e>

October 30th, email including a set of questions about the API and a response.

“Hi Kim, Jason:

I’m glad you asked this question, since I’ve gotten to the point of having some running code in the kind of API you requested and I have some questions of my own as to how the final approach should look.   I’ve outlined the approach in a number of meetings but it will be useful to record this and match it to your original request.

>search will still be performed through LexEVS CNG methods and then somehow all matched nodes will be

>identified through graph db and constructed to form an iterator (ResolvedReferenceConceptIterator) with precision very

>efficiently

The search is being performed through the text searches available in the CodedNodeSet API and matched to the methods you requested your original note referencing the 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)

>As a related question, if you plan to turn the graph db implementation into a microservice, what would be the

>scope or the footprint of this microservice?

The code review we did of this was of a microservice that would be available to users.  As I noted in the code review 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 for a given iterator.

>Would it include the LexEVS CNG APIs

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

> Can it be easily integrated with the SPARQL

work in EVSRESTAPI?

I don’t know EVSRESTAPI well enough to comment on its integration.  We can create a JIRA item to investigate if you like.   We currently load the Arango graph database from LexEVS.  Depending on how complicated it is to do the same kind of work using SPARQL, I see no reason we can perform the same load from a triple store.

I don’t know much about the nature of our meeting with Harold, so I don’t know if we should discuss then or not.  -scott”

November 5th, We review and discuss the API at the Architectural Meeting

[https://wiki.nci.nih.gov/display/LexEVS/2019-11-05+Meeting+notes](https://wiki.nci.nih.gov/display/LexEVS/2019-11-05%2BMeeting%2Bnotes)

November 7th The documented API is committed to the code base. This documentation describes how the API is supposed to work.

<https://github.com/lexevs/lexevs/commit/57e53f533c684650c31154c963b1685f80753f02>

November 12th, We hold an hour-long code review of the API, which at this point is fully documented in the code base.

November 22nd, We hold an hour long code review of changes worked in as the result of the November 12th code review.