LexEVS 6.0 Design Document - Detailed Design - XML Export

Document Information

Author: Craig Stancl
Email: Stancl.craig@mayo.edu
Team: LexEVS
Contract: CBITT BOA Subcontract# 29XS223
Client: NCI CBIIT
National Institutes of Heath
US Department of Health and Human Services

Revision History

Version	Date	Description of Changes	Author
1.0	5/14/10	Initial Version Approved via Design Review	Team

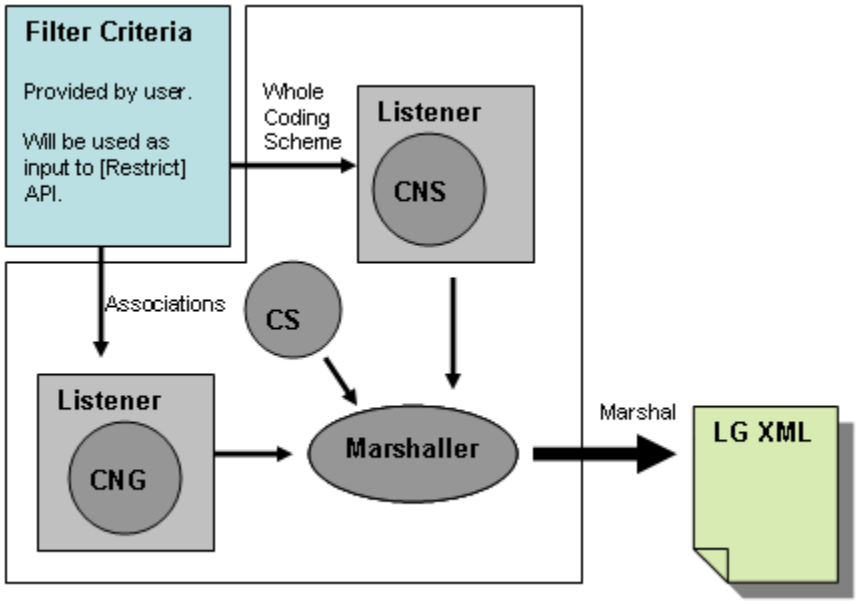
XML exporter exports the LexEVS 6.0 contents (coding scheme and components) from LexEVS database to an XML file. The output XML file will be a valid LGXML file. The process of extracting content from LexEVS is done by using LexEVS 6 APIs.

The exporter will also have the ability to export a subset of coding scheme content. Users will be able to provide criteria that will determine what content is returned. An example would be to export only associations.

The exporter will stream the content from LexEVS to the XML file. This will allow for large ontologies to be exported.

Right now the design to accomplish this is as follows:

1. Read the CodingScheme object.
2. Get CodedNodeSet/CodedNodeGraph from the coding scheme.
3. Add a dummy entity to the coding scheme object.
4. Add Entities (or associations) by applying the filter criteria -
    a. In the first phase of development two uses cases will be supported:
        i. Export the whole coding scheme
        ii. Export only the associations of a coding scheme.
    b. Later, a richer set of user supplied criteria will be supported.
5. Construct a marshaller that will marshal the select contents in step 4.
6. Create a listener that will take the CodedNoteSet, CodedNodeGraph and output file details as parameters.
7. Add the listener to the marshaller.
8. Marshal the coding scheme with the created marshaller. During the marshalling processes the listener will call into LexGrid to get contents to further marshal. For example, it will detect our dummy entity object and make a call to LexEVS to get a block of entity objects.
9. The marshaller and listener will work together to stream the contents to the output file.



Two Listeners are shown in the flow diagram. These may be combined into a single listener that will contain both the CodedNodeGraph and CodedNodeSet objects.

In Addition to this we plan to develop JUnits and other auxiliary set of resources that will be make it ready to be hooked up to DOA layer when ready.

Finally, documentation will be provided to explain how to supply filtering criteria to the exporter.