LexEVS 6.0 Functionality Overview

The following information was designed to give an overview of the functionality contained within the Lexical Grid Enterprise Vocabulary Services (LexEVS). As a beginner or experienced terminologist you will find this information useful when choosing if the LexEVS services are right for your institution or application.

Each area of LexEVS is broken down with its primary value proposition and additional items for you to consider.

Service Area	Main Functions	Additional Functional Qualification
Model - The LexGrid model is Mayo Clinic's proposal for standard storage of controlled vocabularies and ontologies. These are the high-level objects incorporated into the model definition.	• Coding System - each vocabulary is modeled as an individual Code System, which could be seen as a container for concepts, relationships, properties and meta data. • Entity - each entity could be of type 'Concept', 'Instance', 'Association' etc. • Concept - subclass of Entity. • Association - subclass of Entity, and represents the relationship between Concepts. • Instance - subclass of Entity. • Property - association with the Coding System, Entity or Association. • Value Set Definition - the contents are coded entities defined in referencing Code System. Value Set can contain coded entities from one or more Code System. • Pick List Definition - defines an ordered list of entity codes and preferred designations drawn from a resolved Value Set Definition.	• Supports synonyms. • Alternate definitions are available. • Supports property qualifiers, however, properties cannot have properties. • Supports relationships of multiple types. • Provides Code System metadata. • Mapping relationship, including "translations" between terminologies. • Supports user-defined properties. • Supports translation or terms with another associated human language. • Supports Subset (concepts related via "is-a" in downward direction). • Supports Subset (concepts related via an explicit set relationship). • Relationships are managed as concepts, for example they have names. • Create subsets with features such as hierarchy, sequencing, grouping, history, and language.
Authoring - The APIs provide the capability to author a Code System and its contents.	• Create new Code System, Code System Property, Concept, Concept Property, Association (add Qualifiers to the Association), and supplement. • Edit Code System meta data, Code System Property, Concept meta data, Concept Property and Association type. • Remove entire Code System, Code System Property, Concept, Concept Property. • Change status of Code System or Concept, such as retiring. • Allows designation of the officially supported Code System, Concept, or Value Set. • Determine source and target Code System for an Association. • New Code System consisting of mappings of one system's concept to another. • New mapping for an existing mapping scheme.	• The LexEVS user interface supports efficient manual use for query and test operations even when operating on large data sets. • A Code System or concept authoring user interface is not available. • A developers GUI for Value Set authoring is available.
Versioning - An ordered collection of state changes that define the transformation of a set of resources from one consistent state to another.	• Provides release information about content loaded into the LexEVS system. • Versionable Entities are the resources that can undergo change over time while maintaining its identity. All the LexGrid elements that can undergo changes such as Entity, CodingScheme, ValueSetDefinition, property, etc. are Versionable Entities.	• The version of loaded content can be checked via APIs, command line, or GUI.
CTS - Comprehensive support for Common Terminology Services.	• CTS defines the functional requirements of a set of service interfaces to allow the representation, access, and maintenance of terminology content either locally, or across a federation of terminology service nodes.	• Fully compliant with CTS 1 and soon CTS 2. • CTS 2 is currently a draft standard. As changes are approved to the standard, LexEVS will implement these changes.
Mapping - Concept, Instance, or Association mapping between two different ontologies.	• LexEVS provides inter-terminology relationship support, authoring, and loading.	• Mapping scheme is loaded independent of the mapped ontologies.
Value Set - Value Set Definition within the LexGrid logical model defines the contents of a Value Set. The contents are coded entities defined in the referenced Code System.	• Administration - ability to load, export, and remove a Value Set Definition. • Query - ability to apply user restrictions and dynamically resolve the definitions at run time. • Resolve - ability to resolve a Value Set Definition dynamically against selected Code System version and return all Concepts belonging to the Value Set.	• Value Sets can contain coded entities from one or more Code Systems. • A developer's Value Set and Pick List user interface is available.
Pick List - Pick List Definition within the LexGrid logical model defines an ordered list of entity codes and corresponding presentations drawn from a resolved Value Set Definition.	• Administration - ability to load, export and remove Pick List Definitions. • Query - ability to apply user restrictions and dynamically resolve the definitions at run time. • Resolve - ability to resolve Pick List Definition dynamically against a selected Code System version and return all concepts or selected concepts belonging to the Pick List.	• Pick Lists can be defined using exclusion or inclusion criteria. • A developer's Value Set and Pick List user interface is available. • Refine Pick Lists by constraints such as the human language or the context of the concept.
Loaders - LexEVS includes the loaders listed in this document.	• NCI MetaThesaurus Loader • OBO loader • OWL loader • Text loader • UMLS loader • MetaData loader • NCI history loader • OBO history loader • ICD9 loader • ICD10 loader • ICD general equivalence mapping loader	• Create custom loaders using the Loader Framework extension.
Exporters - LexEVS includes predefined exporters.	• LexGrid XML exporter with filter functionalities • OBO exporter • OWL/RDF exporter	• Supports the versioning of concepts, value sets, and terminologies. • Create custom exporters using the Exporter Framework extension.

Query - LexEVS provides a set of query services and APIs	• By Entity - Code Node Set. • By association - Code Node Graph. • Concept Domain Query API - provides capability to query Concept Domain available in the LexEVS system and also to query the binding between Value Set and Concept Domain. • Usage Context Query API - provides capability to query Usage Context available in the LexEVS system and also to query the binding between Value Set and Concept Domain in conjunction with Usage Context. • Value Set Query API - provides capability to query Value Sets available in the LexEVS system and also to query the binding between Value Set and concept domain.	• Provides answers to queries concerning the LexEVS system capabilities, terminologies supported, relationship types, search algorithms, and versions. • Provides answers to queries concerning terminology properties. • Supports search based on all concept properties and relationships. • Supports a variety of robust and varied methods for querying the LexEVS system. • Presents more than one concept at time for comparison purposes. • Supports search based on any of a concept's properties or relationships.
Web services - LexEVS provides a couple of web services.	• SOAP • REST	• Prototype functionality provided prior to and including LexEVS 6.0. • LexEVS 6.1 will broaden and harden these interfaces.

Extensibility - LexEVS provides extendability through a pluggable framework for use locally or as contributions to the community via open source.	• The LexEVS APIs, the core APIs, fall into three primary categories:
	  ◦ Core services - Includes the LexBIGService, LexBIGServiceManager, CodedNodeSet and CodedNodeGraph classes, which provide the initial entry points for programmatic access to all LexEVS system features and data.
	  ◦ Service extensions - The extension mechanism provides for pluggable features. Current extension points allow for the introduction of custom load and indexing mechanisms; unique query, sort, and filter mechanisms; and generic functional extensions which can be advertised for availability to client programs.
	  ◦ Utilities - Utility classes, such as those implementing iterator support, are provided by the LexEVS system to provide convenience and optimize the handling of resources accessed through the LexEVS runtime.
	• caCORE Data Services API: it is a public domain, open source wrapper that provides full access to the LexEVS Terminology Server. LexEVS hosts the NCI Thesaurus, the NCI Metathesaurus, and several other vocabularies.
	• Analytical Grid Service API: uses a version of the LexGRID/LexBIG model, extended to support ISO 21090 Datatypes, and fits for Grid Client Applications.
	• Data Grid Services API: is a standard caGrid Data service. See caGrid Data Service Documentation and caGrid Data Service API Documentation for more information.
	• Preliminary CTS2 API: provides programmatic access to LexEVS 6.0 implementation of CTS 2 features and services. Developers can build custom applications, tools, and services that can make calls as per CTS 2 specification against LexEVS CTS 2 API. Visit Common Terminology Services 2 for services description, purpose and scope of CTS 2 specification.
		• LexEVS uses an open architecture supporting extensions created by external organizations.
		• The LexEVS code base and documentation of algorithms (including description logic) is open source to support maintenance and extensibility.

Semantic Web - LexEVS supports the semantic web.	• OWL/RDF loader and API. • OWL/RDF exporter and API. • OWL/RDF exporter can export an ontology to a triple store, based on Jena 2.6.3.	• Import and export functions can be executed via APIs, command line, or GUI.
Other - Relevant non-functional attributes are typically covered by the underlying architecture.	• Security • Availability • Backup and Recovery • Scalability and performance • License	• Custom security implementation configurable per Coding System in a distributed environment. • Install instructions contain recommended approaches for certain database management systems. • Conforms to licensing needs.