

# LexEVS 6.x Information Models

## Contents of this Page

- [Information Models Overview](#)
- [LexGrid Model](#)
- [CodingSchemes](#)
- [Entities](#)
  - [Entity](#)
  - [Relations](#)
  - [AssociationEntity](#)
- [Value Set Definition](#)
- [Pick List Definition](#)
- [Naming](#)
- [Versions](#)
  - [SystemRelease](#)
  - [VersionableEntities](#)
  - [Versioning](#)
- [LexBIG model](#)
- [Collection](#)
- [Core](#)
- [Enums](#)
- [InterfaceElements](#)
- [NCIHistory](#)

## LexEVS 6.x Design Links to Include

- [Design Guide Main Page](#)
  - [LexEVS Information Models](#)
  - [LexEVS Architecture](#)
- [LexEVS 6.0 Main Page](#)
- [LexEVS Current Release](#)

## Information Models Overview

The information below is provided for introductory purposes. A full description of all available model components is also available in the javadoc distributed with the LexEVS installation package (see file breakdown in the [LexEVS 6.x Installation Guide](#)). Since the javadoc is automatically generated and synchronized during the build process, it is recommended as the primary reference for use by LexEVS developers.

## LexGrid Model

The LexGrid Model is Mayo Clinic's proposal for standard storage of controlled vocabularies and ontologies. The LexGrid Model defines how vocabularies should be formatted and represented programmatically, and is intended to be flexible enough to accurately represent a wide variety of vocabularies and other lexically-based resources. The model also defines several different server storage mechanisms and an XML format. This model provides the core representation for all data managed and retrieved through the LexEVS system, and is now rich enough to represent vocabularies provided in numerous source formats such as OWL (NCI Thesaurus) and RRF (NCI MetaThesaurus).


Once the vocabulary information is represented in a standardized format, it becomes possible to build common repositories to store vocabulary content and common programming interfaces and tools to access and manipulate that content. The LexBIG API developed for caBIG® is one such interface, and is described in additional detail in LexEVS APIs.

The LexGrid model is mastered in XML schema. The LexEVS project currently builds on the 2010 version of the LexGrid schema. A complete list of XML schemas, XSLT converters, model in Enterprise Architect (EA) format are available at [LexGrid Model and Schema](#).

Following are some of the higher-level objects incorporated into the model definition:



## CodingSchemes

Each service defined to the LexGrid model can encapsulate the definition of one or more vocabularies. Each vocabulary is modeled as an individual code system, known as a codingScheme. Each scheme tracks information used to uniquely identify the code system, along with relevant metadata and is a high level container for entities and relations. Each CodingScheme represents a unique code system or version in the LexEVS service. The collection of all code systems defined to a service is encapsulated by a single codingSchemes container.

- **Schema:** [CodingSchemes.xsd](#) 
- **Enterprise Architect (EA) format:** [CodingScheme model](#) . Click on any [attributes](#)  to view more details.


## Entities

A code system may define zero or more coded entities, encapsulated within a single container. An entity represents a coded entity (identified in the model as a entity) within a particular domain of discourse. Each entity could be of type 'concept', 'instance', 'association' etc., and is unique within the code system that defines it. To be valid, an entity must be qualified by at least one designation, represented in the model as a property. Each property is an attribute, facet, or some other characteristic that may represent or help define the intended meaning of the encapsulating entity. An entity may be the source for and/or the target of zero or more relationships. Relationships are described in more detail in a following section. The collection of all coded entities defined with in a coding scheme is encapsulated by a single entities container.

- **Schema:** [Entities.xsd](#) 
- **Enterprise Architect (EA) format:** [Entities model](#) . Click on any attributes to view more details.

## Entity



Entity is a set of lexical assertions about the intended meaning of a particular entity code. Each entity represents a unique entity within the code system, it could be of type 'concept', 'instance', 'association' etc., and can be further described by properties and its relation to other entities through relations. The collection of all coded entities defined with in a coding scheme is encapsulated by a single entities container.

- **Schema:** [Entities.xsd](#) 
- **Enterprise Architect (EA) format:** [Entity model](#) . Click on any attributes to view more details.

## Relations



Relations are used to define and qualify associations between entities. Each code system may define one or more containers to encapsulate relationships between entities. Each named relationship (e.g. "hasSubtype" or "hasPart") is represented as an association within the LexGrid model. Each relations container must define one or more association. The association definition may also further define the nature of the relationship in terms of transitivity, navigable, forward and inverse names, etc. Multiple instances of each association can be defined, each of which provide a directed relationship between one source and one or more target entities.

Source and target entities may be contained in the same code system as the association or another if explicitly identified. By default, all source and target entities are resolved from the code system defining the association. The code system can be overridden by each specific association, relation source (associationInstance), or relation target (associationTarget).

- **Schema:** [Relations.xsd](#) 
- **Enterprise Architect (EA) format:** [Relations model](#) . Click on any attributes to view more details.

## AssociationEntity

AssociationEntity is a subclass of an Entity with type as 'association'. AssociationEntity basically defines the nature of the relationship in terms of transitivity, navigable, forward and reverse names.

- **Schema:** [Relations.xsd](#) 
- **Enterprise Architect (EA) format:** [AssociationEntity model](#) . Click on any attributes to view more details.

## Value Set Definition

Value Set Definition with in the LexGrid logical model defines the contents of a Value Set. The contents are coded entities defined in referencing Code System. Value Set can contain coded entities from one or more Code Systems.

Visit [LexEVS 6.x Value Set Detailed Design](#) for detailed information about Value Set Definition.

- **Schema:** [ValueSets.xsd](#) 
- **Enterprise Architect (EA) format:** [Value Set Definition model](#) . Click on any attributes to view more details.

## Pick List Definition

Pick List Definition with in the LexGrid logical model defines an ordered list of entity codes and corresponding presentations drawn from a resolved value set definition.


Visit [LexEVS 6.x Pick List Detailed Design](#) for detailed information about Pick List Definition.

- **Schema:** [ValueSets.xsd](#) 
- **Enterprise Architect (EA) format:** [Pick List Definition model](#) . Click on any attributes to view more details.

## Naming

These are the elements primarily used to define metadata for a coding scheme, value set and pick list definition, mapping locally used names to global references. Each naming elements are represented as SupportedXXX where XXX could be language, property name, source, context etc., that are locally used in coding scheme, value set definition or pick list definition. It also has an optional URI that can be used to find the exact definition and meaning of the local id. Note: the string portion of this entry can be used to provide additional documentation or information, especially when a URI is not supplied.

- **Schema:** [Naming.xsd](#) 



- **Enterprise Architect (EA) format:** [Naming model](#)  . Click on any attributes to view more details.

## Versions

These are the elements primarily used to author terminology elements.

## SystemRelease

System Release is a collection of coding schemes, value set definitions, pick list definitions and/or revision records that are distributed as a unit.

- **Schema:** [Versions.xsd](#) 
- **Enterprise Architect (EA) format:** [System Release model](#)  . Click on any attributes to view more details.


## VersionableEntities

Versionable Entities are the resources that can undergo change over time while maintaining its identity. All the LexGrid elements that can undergo changes (like Entity, CodingScheme, ValueSetDefinition, property etc) are Versionable Entities.


- **Schema:** [Versions.xsd](#) 
- **Enterprise Architect (EA) format:** [Versionable Entities model](#)  . Click on any attributes to view more details.

## Versioning

Versioning are an ordered collection of state changes that define the transformation of a set of resources from one consistent state to another.

- **Schema:** [Versions.xsd](#) 
- **Enterprise Architect (EA) format:** [Versioning model](#)  . Click on any attributes to view more details.

## LexBIG model

The following extensions to the LexGrid model were introduced in support of caBIG® requirements. As with the LexGrid model, this document provides a summary of the most significant elements for consideration by LexBIG programmers. The complete and current version of the model is [available online](#) .

The LexBIG logical model consists of following five XML schemas:

- **Collection:** Contains named sets of things.
- **Core:** Contains the core elements of LexBIG model.
- **\*Enums:** Enums for various LexBIG model elements.
- **Interface Elements:** Defines metadata related to model objects required by the runtime.
- **NCI History:** Version change definition for NCI change model.

## Collection

LexBIG Collection contains named sets of things.

- **Schema:** Collection.xsd:

```
http://lexgrid.org/LexGrid/downloads/LexBIG/schemas/2010/01/Collection.xsd
```

- **Enterprise Architect (EA) format:** [Collection model](#)  . Click on any attributes to view more details.

## Core

LexBIG core elements provide enhanced referencing and controlled resolution of LexGrid model objects.


- **Schema:** [Core.xsd](#) 
- **Enterprise Architect (EA) format:** [Core model](#)  . Click on any attributes to view more details.

## Enums

LexBIG Enums contains various enums available in LexBIG model.



- **Schema:** Enums.xsd:

```
http://lexgrid.org/LexGrid/downloads/LexBIG/schemas/2010/01/Enums.xsd
```

- **Enterprise Architect (EA) format:** [Enums model](#)  . Click on any attributes to view more details.



## InterfaceElements

Defines metadata related to model objects required by the runtime.

- **Schema:** [InterfaceElements.xsd](#) 
- **Enterprise Architect (EA) format:** [Interface Elements model](#)  . Click on any attributes to view more details.

## NCIHistory

Maintains a record of modifications made to a code system.

- **Schema:** [NCIHistory.xsd](#) 
- **Enterprise Architect (EA) format:** [NCI History model](#)  . Click on any attributes to view more details.