National Cancer Institute
caBIG Knowledge Center
A part of the Enterprise Support Network
U.S. National Institutes of Health | www.cancer.gov

- Home
- Knowledge Centers
  - caGrid
  - Clinical Trials Management Systems
  - Data Sharing and Intellectual Capital
  - Molecular Analysis Tools
  - Tissue/Biospecimen Banking and Technology Tool
  - Vocabulary
- Discussion Forums
  - caGrid
  - Clinical Trials Management Systems
  - Data Sharing and Intellectual Capital
  - Molecular Analysis Tools
  - Tissue/Biospecimen Banking and Technology Tool
  - Vocabulary
- Bugs/Feature Requests
- Development Code Repository

LexEVS 5.0 Design and Architecture Guide

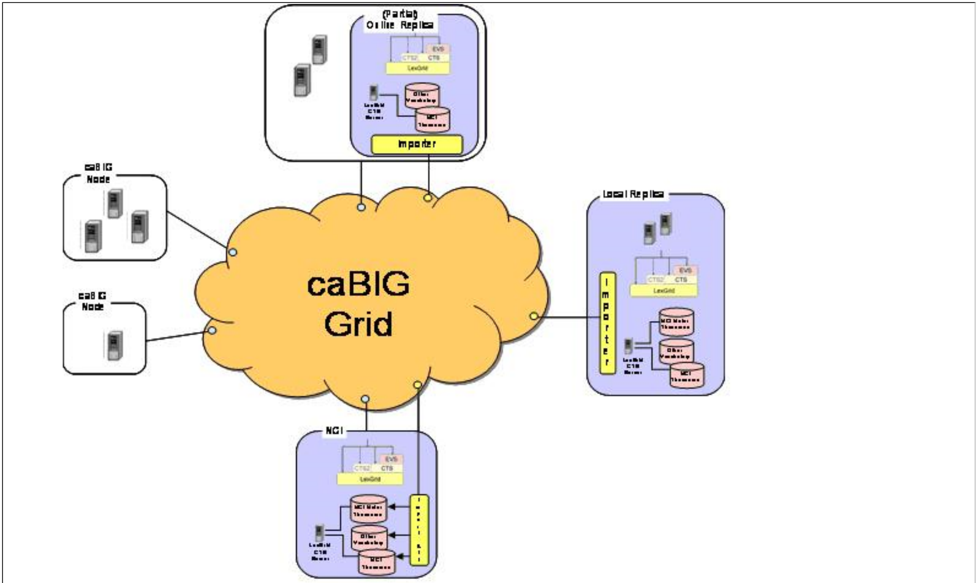
From Vocab_Wiki

LexEVS New Documentation > LexBig and LexEVS > LexEVS Version 5.0 > LexEVS 5.0 Documentation > LexEVS 5.0 Design and Architecture Guide

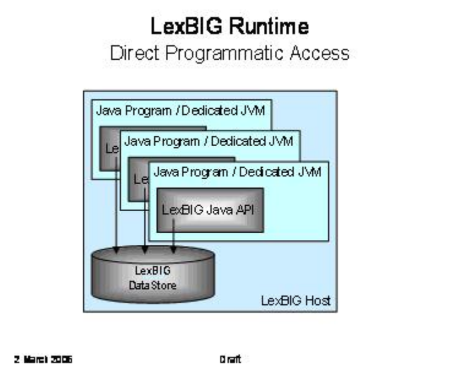
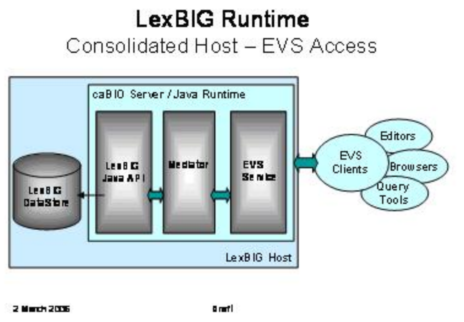
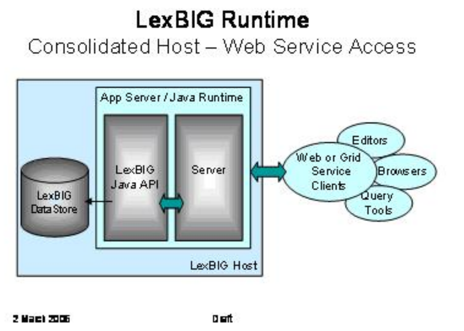
Contents

Overview

LexBIG software architecture and implementation is designed to facilitate flexibility and future expansion. The following diagrams are intended to aid the understanding of LexBIG service integration in context of the larger caBIG® universe and specific deployment scenarios:



This diagram depicts the LexBIG vision. Individual Cancer Centers will be able to use the existing set of caCORE EVS services. If desired, local instances of vocabularies can be installed.

 LexBIG Runtime Direct Programmatic Access Java Program / Dedicated JVM Java Program / Dedicated JVM Java Program / Dedicated JVM LexBIG Java API LexBIG Data Store LexBIG Host 2 March 2006 Draft	This diagram depicts direct Java-to-Java access to LexBIG functions. This is the primary deployment scenario for phase 1. Note: It is not required that the database be located on the same system as the program runtime.
 LexBIG Runtime Consolidated Host – EVS Access caBIO Server / Java Runtime LexBIG DataStore LexBIG Java API Mediator EVS Service EVS Clients Editors Browsers Query Tools LexBIG Host 2 March 2006 Draft	This diagram depicts access through caCORE Enterprise Vocabulary Services (EVS) to a LexBIG vocabulary engine. The primary goal is to provide a compatible experience for existing EVS browsers and client applications. Note: this diagram shows the possible inclusion of a mediation layer between EVS and the LexBIG runtime. This would be done to facilitate alternate communications with the LexBIG server (e.g. through web services as described below).
 LexBIG Runtime Consolidated Host – Web Service Access App Server / Java Runtime LexBIG Data Store LexBIG Java API Server Web or Grid Service Clients Editors Browsers Query Tools LexBIG Host 2 March 2006 Draft	The LexBIG API is designed with web and grid-level enablement in mind. This diagram depicts deployments that wrap the current API to allow the runtime to be accessed through web or grid services.

What is LexGrid?

LexGrid is an initiative of the Mayo Clinic Division of Biomedical Informatics that focuses on the representation, storage, and dissemination of vocabularies. This effort centers on, but is not limited to, the domain of medical vocabularies and nomenclatures. Focal points of the LexGrid project include the development and promotion of standards, tools, and content that:

- Provide flexibility to represent yesterday’s, today’s and tomorrow’s terminological resources using a single information model.
- Provide the ability for these resources to be published online, cross-linked, and indexed.
- Provide standardized building blocks and tools that allow applications and users to take advantage of the content where and when it is needed.
- Provide consistency and standardization required to support large-scale terminology adoption and use.

Additional information for LexGrid is available at http://informatics.mayo.edu .

What is LexBIG?

LexBIG is a more specific project that applies LexGrid vision and technologies to requirements of the caBIG® community. The goal of the project is to build a vocabulary server accessed through a well-structured application programming interface (API) capable of accessing and distributing vocabularies as commodity resources. The server is to be built using standards-based and commodity technologies. Primary objectives for the project include:

- Provide a robust and scalable open source implementation of EVS-compliant vocabulary services. The API specification will be based on but not limited to fulfillment of the caCORE EVS API. The specification will be further refined to accommodate changes and requirements based on prioritized needs of the caBIG® community.
- Provide a flexible implementation for vocabulary storage and persistence, allowing for alternative mechanisms without impacting client applications or end users. Initial development will focus on delivery of open source freely available solutions, though this does not preclude the ability to introduce commercial solutions (e.g. Oracle).
- Provide standard tooling for load and distribution of vocabulary content. This includes but is not limited to support of standardized representations such as UMLS Rich Release Format (RRF), the OWL web ontology language, and Open Biomedical Ontologies (OBO) .

The goal for the initial year of development was to achieve the Bronze level of compatibility with regard to the caBIG® requirements. Silver-level compatibility is being pursued.

LexGrid Model

The LexGrid Model is Mayo’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 a XML format. This model provides the core representation for all data managed and retrieved through the LexBIG 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 LexBIG APIs.

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

Code Systems

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. The collection of all code systems defined to a service is encapsulated by a single *codingSchemes* container.

Concepts

A code system may define zero or more coded concepts, encapsulated within a single container. A concept represents a coded entity (identified in the model as a *concept*) within a particular domain of discourse. Each concept is unique within the code system that defines it. To be valid, a concept 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 concept. A concept may be the source for and/or the target of zero or more relationships. Relationships are described in more detail in a following section.

Relations

Each code system may define one or more containers to encapsulate relationships between concepts. 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, symmetry, reflexivity, 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 concepts.

Source and target concepts may be contained in the same code system as the association or another if explicitly identified. By default, all source and target concepts 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*).

LexBIG Extensions

The LexBIG vocabulary model extends the LexGrid model to provide unique constructs or granularity required by caBIG® that are not present in the core model. While many extensions exist, this document will focus on some of direct relevance to the high-level architecture.

Concept Resolution

LexBIG allows the service runtime to provide managed resolution of code-based objects that are referenced through LexBIG-specific lists and iterators (mechanism that allow streaming of list content). These lists and iterators are typically returned when requesting sets or graphs of vocabulary terms through the LexBIG API (described in LexBIG APIs). Some model components involved in the resolution process include:

`ConceptReference` – A globally unique reference to a concept code.

`ResolvedConceptReference` - A concept reference for which additional information has been resolved, including description and relationship participation.

`AssociatedConcept` - A concept reference that contains full detail in participation as a source or target of an association, including indications of navigability and qualification.

Note: Formal representation of the LexGrid and LexBIG models are discussed in Information Models.

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 5.0 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 mastered in XML Schema. The LexBIG project currently builds on the 2008 version of the LexGrid schema. A formal representation, showing portions of this structure that are of primary interest to the LexBIG project, is presented below. A complete version of the model is available at http://informatics.mayo.edu?page=lgm .

CodingSchemes

The CodingSchemes branch of the model defines high level containers for concepts and relations. Each CodingScheme represents a unique code system or version in the LexBIG service. Components of interest include:

codingScheme

codingSchemes

A collection of one or more coding schemes.

codingScheme

A resource that makes assertions about a collection of terminological entities.

entities

A set of entity codes and their lexical descriptions

relations

A collection of relations that represent a particular point of view or community.

versions

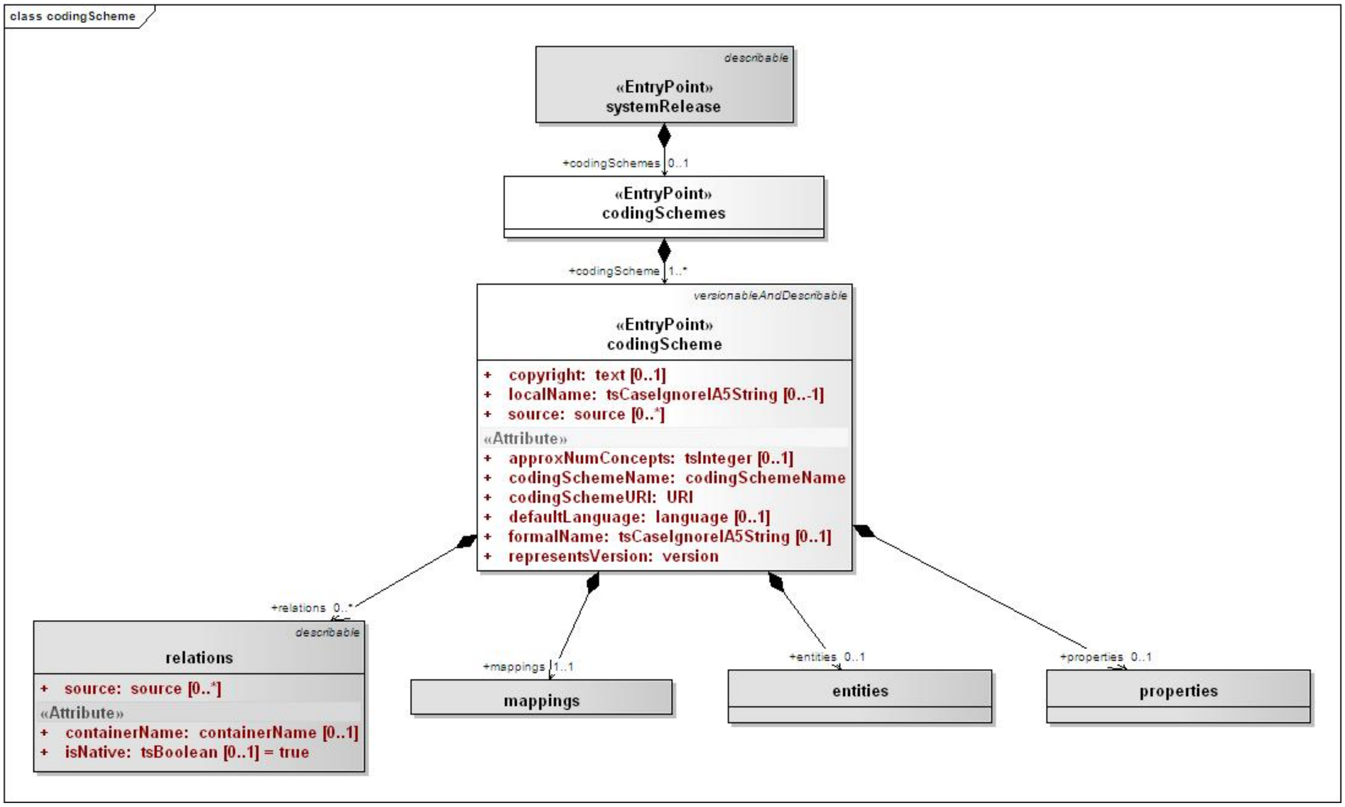
A list of past versions of the coding scheme.

mappings

A list of all of the local identifiers and defining URI's that are used in the associated resource

properties

A collection of properties.



codingSchemes

Concepts

Each concept represents a unique entity within the code system, which can be further described by properties and related to other concepts through relations.

conceptsAndInstances

codingScheme

A resource that makes assertions about a collection of terminological entities.

entities

A set of entity codes and their lexical descriptions

entity

A set of lexical assertions about the intended meaning of a particular entity code.

concept

An entity that represents a class or category. The entityType for the class concept must be "concept".

instance

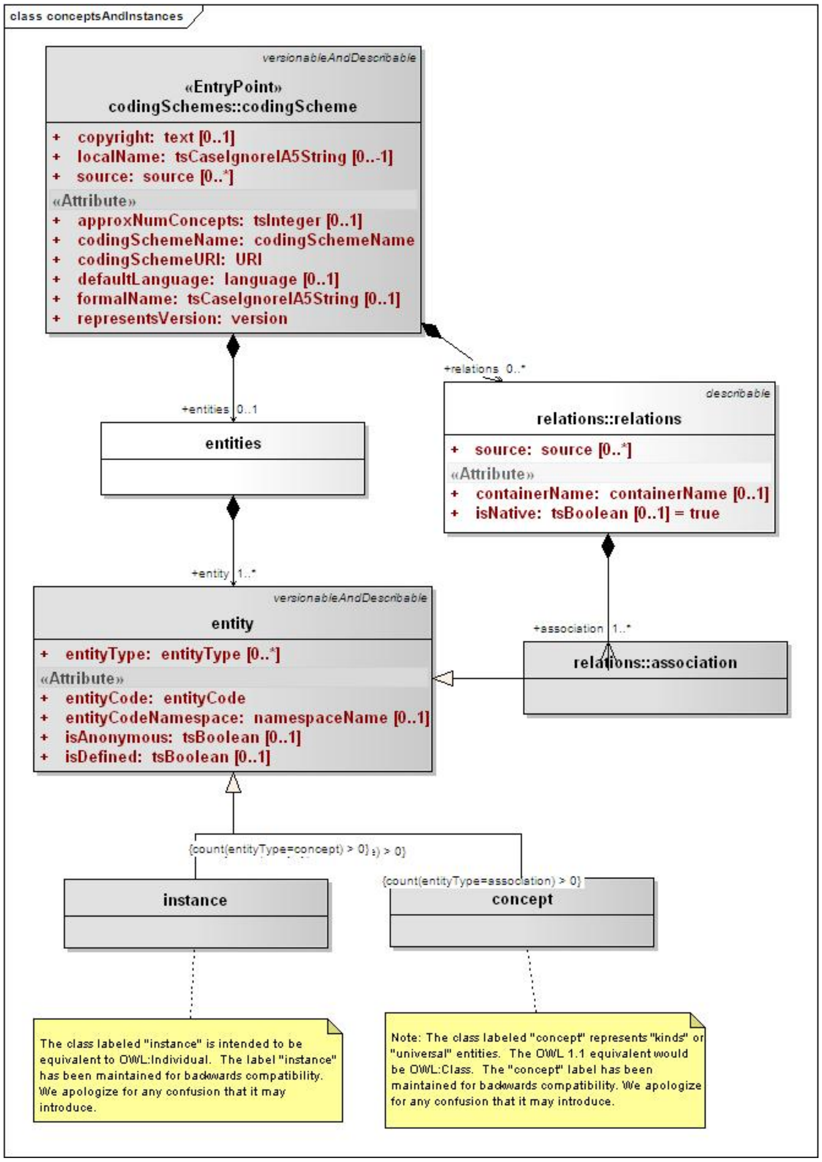
An entity that represents an instance or an individual. The entityType for the class concept must be "instance".

relations

A collection of relations that represent a particular point of view or community.

association

A binary relation from a set of entities to a set of entities and/or data. The entityType for the class concept must be "association".


class conceptsAndInstances
versionableAndDescribable
«EntryPoint»
codingSchemes::codingScheme
+ copyright: text [0..1]
+ localName: tsCaseIgnoreIA5String [0..-1]
+ source: source [0..*]
«Attribute»
+ approxNumConcepts: tsInteger [0..1]
+ codingSchemeName: codingSchemeName
+ codingSchemeURI: URI
+ defaultLanguage: language [0..1]
+ formalName: tsCaseIgnoreIA5String [0..1]
+ representsVersion: version
+entities 0..1
entities
+relations 0..*
describable
relations::relations
+ source: source [0..*]
«Attribute»
+ containerName: containerName [0..1]
+ isNative: tsBoolean [0..1] = true
+entity 1..*
versionableAndDescribable
entity
+ entityType: entityType [0..*]
«Attribute»
+ entityCode: entityCode
+ entityCodeNamespace: namespaceName [0..1]
+ isAnonymous: tsBoolean [0..1]
+ isDefined: tsBoolean [0..1]
+association 1..*
relations::association
{count(entityType=concept) > 0}
{count(entityType=association) > 0}
instance
concept
The class labeled "instance" is intended to be equivalent to OWL:Individual. The label "instance" has been maintained for backwards compatibility. We apologize for any confusion that it may introduce.
Note: The class labeled "concept" represents "kinds" or "universal" entities. The OWL 1.1 equivalent would be OWL:Class. The "concept" label has been maintained for backwards compatibility. We apologize for any confusion that it may introduce.

conceptsAndInstances

entities

codingScheme

A resource that makes assertions about a collection of terminological entities.

entities

A set of entity codes and their lexical descriptions

entity

A set of lexical assertions about the intended meaning of a particular entity code.

concept

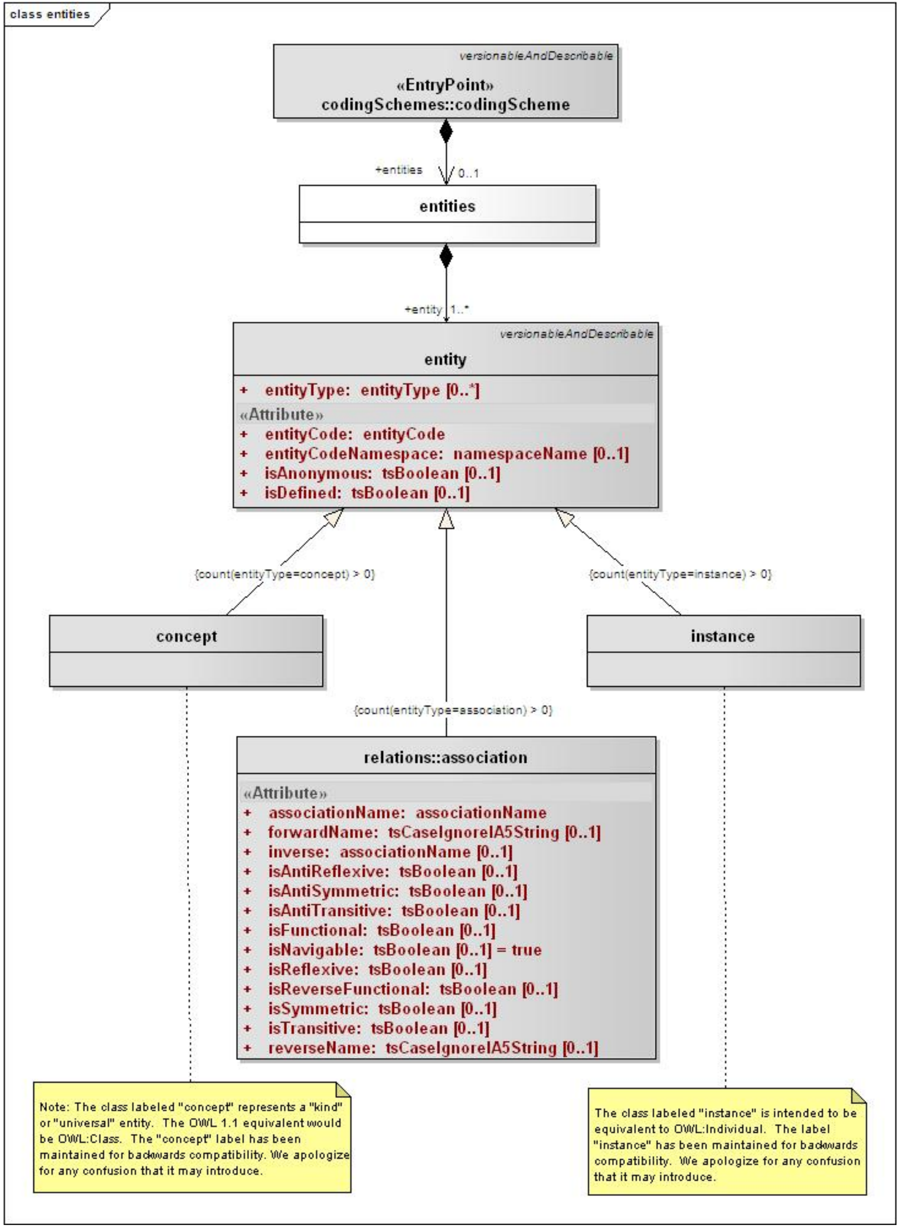
An entity that represents a class or category. The entityType for the class concept must be "concept".

instance

An entity that represents an instance or an individual. The entityType for the class concept must be "instance".

association

A binary relation from a set of entities to a set of entities and/or data. The entityType for the class concept must be "association".


class entities
versionableAndDescribable
«EntryPoint»
codingSchemes::codingScheme
+entities
0..1
entities
+entity
1..*
versionableAndDescribable
entity
+ entityType: entityType [0..*]
«Attribute»
+ entityCode: entityCode
+ entityCodeNamespace: namespaceName [0..1]
+ isAnonymous: tsBoolean [0..1]
+ isDefined: tsBoolean [0..1]
{count(entityType=concept) > 0}
{count(entityType=instance) > 0}
concept
instance
{count(entityType=association) > 0}
relations::association
«Attribute»
+ associationName: associationName
+ forwardName: tsCaseIgnoreIA5String [0..1]
+ inverse: associationName [0..1]
+ isAntiReflexive: tsBoolean [0..1]
+ isAntiSymmetric: tsBoolean [0..1]
+ isAntiTransitive: tsBoolean [0..1]
+ isFunctional: tsBoolean [0..1]
+ isNavigable: tsBoolean [0..1] = true
+ isReflexive: tsBoolean [0..1]
+ isReverseFunctional: tsBoolean [0..1]
+ isSymmetric: tsBoolean [0..1]
+ isTransitive: tsBoolean [0..1]
+ reverseName: tsCaseIgnoreIA5String [0..1]

Note: The class labeled "concept" represents a "kind" or "universal" entity. The OWL 1.1 equivalent would be OWL:Class. The "concept" label has been maintained for backwards compatibility. We apologize for any confusion that it may introduce.

The class labeled "instance" is intended to be equivalent to OWL:Individual. The label "instance" has been maintained for backwards compatibility. We apologize for any confusion that it may introduce.

entities

entity

entity

A set of lexical assertions about the intended meaning of a particular entity code.

comment

A property that is used as an annotation or other note about the state or usage of the entity. The propertyType of comment must be "comment"

definition

A property that defines the entity in a particular langage or context.. The propertyType of definition must be "definition"

presentation

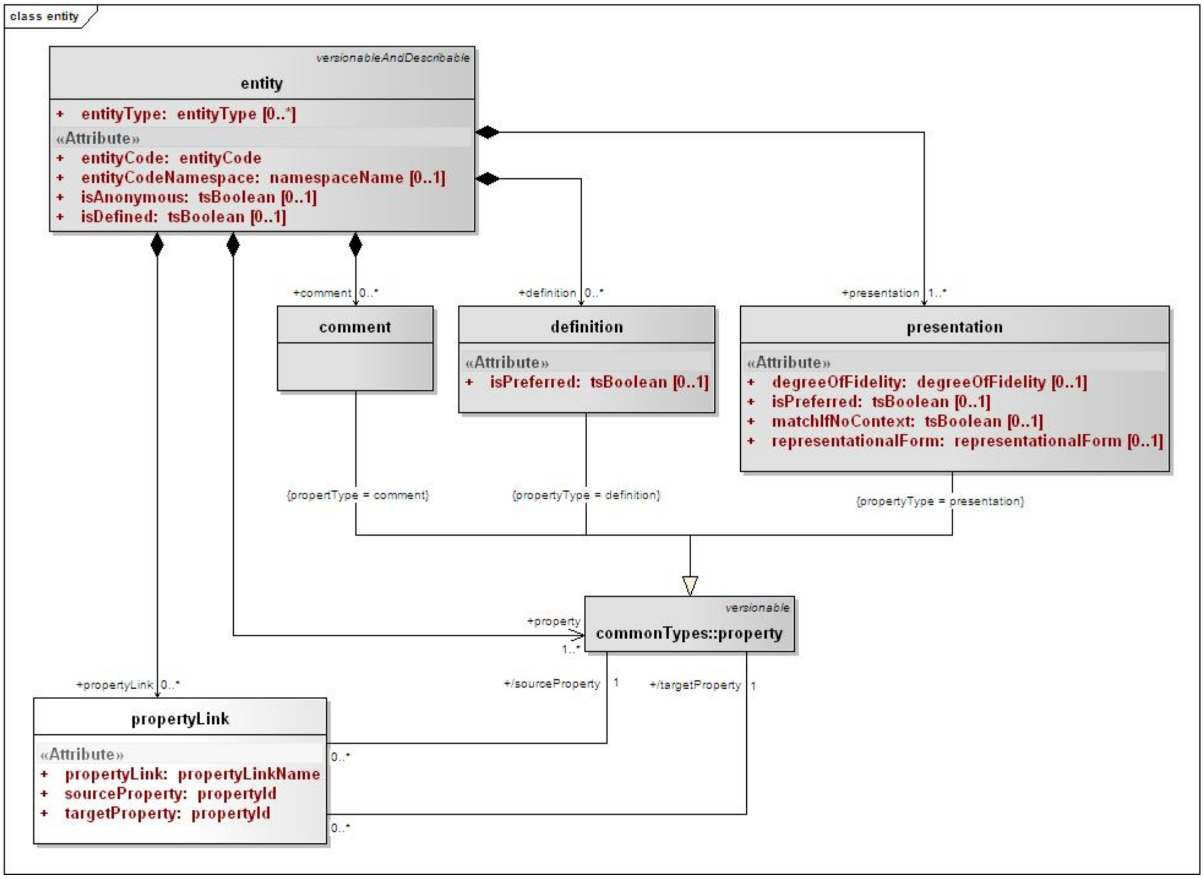
A property ths represents or designates the meaning of the entityCode. The propertyType of presentation must be "presentation"

property

A description, definition, annotation or other attribute that serves to further define or identify an resource.

propertyLink

A link between two properties for an entity.. Examples include acronymFor, abbreviationOf, spellingVariantOf, etc. Must be in supportedPropertyLink.


class entity
versionableAndDescribable
entity
+ entityType: entityType [0..*]
«Attribute»
+ entityCode: entityCode
+ entityCodeNamespace: namespaceName [0..1]
+ isAnonymous: tsBoolean [0..1]
+ isDefined: tsBoolean [0..1]
+comment 0..*
+definition 0..*
+presentation 1..*
comment
definition
«Attribute»
+ isPreferred: tsBoolean [0..1]
presentation
«Attribute»
+ degreeOfFidelity: degreeOfFidelity [0..1]
+ isPreferred: tsBoolean [0..1]
+ matchIfNoContext: tsBoolean [0..1]
+ representationalForm: representationalForm [0..1]
{propertType = comment}
{propertyType = definition}
{propertyType = presentation}
versionable
+property
commonTypes::property
1..*
+/sourceProperty 1
+/targetProperty 1
+propertyLink 0..*
propertyLink
«Attribute»
+ propertyLink: propertyLinkName
+ sourceProperty: propertyId
+ targetProperty: propertyId
0..*
0..*

entity

Relations

Relations are used to define and qualify associations between concepts.

association

codingScheme

A resource that makes assertions about a collection of terminological entities.

relations

A collection of relations that represent a particular point of view or community.

entity

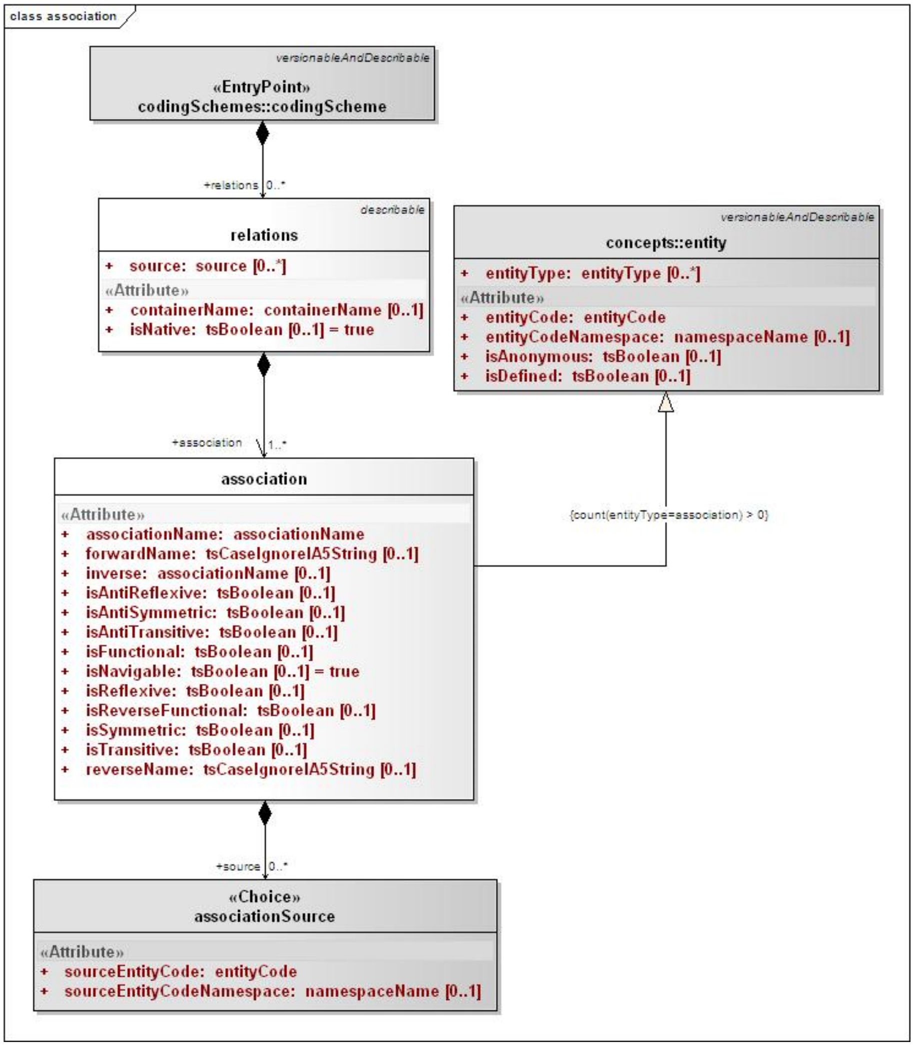
A set of lexical assertions about the intended meaning of a particular entity code.

association

A binary relation from a set of entities to a set of entities and/or data. The entityType for the class concept must be "association".

associationSource

An entity that occurs in one or more instances of a relation on the "from" (or left hand) side of a particular relation.


class association
versionableAndDescribable
«EntryPoint»
codingSchemes::codingScheme
+relations
0..*
describable
relations
+ source: source [0..*]
«Attribute»
+ containerName: containerName [0..1]
+ isNative: tsBoolean [0..1] = true
versionableAndDescribable
concepts::entity
+ entityType: entityType [0..*]
«Attribute»
+ entityCode: entityCode
+ entityCodeNamespace: namespaceName [0..1]
+ isAnonymous: tsBoolean [0..1]
+ isDefined: tsBoolean [0..1]
+association
1..*
association
«Attribute»
+ associationName: associationName
+ forwardName: tsCaseIgnoreIA5String [0..1]
+ inverse: associationName [0..1]
+ isAntiReflexive: tsBoolean [0..1]
+ isAntiSymmetric: tsBoolean [0..1]
+ isAntiTransitive: tsBoolean [0..1]
+ isFunctional: tsBoolean [0..1]
+ isNavigable: tsBoolean [0..1] = true
+ isReflexive: tsBoolean [0..1]
+ isReverseFunctional: tsBoolean [0..1]
+ isSymmetric: tsBoolean [0..1]
+ isTransitive: tsBoolean [0..1]
+ reverseName: tsCaseIgnoreIA5String [0..1]
{count(entityType=association) > 0}
+source
0..*
«Choice»
associationSource
«Attribute»
+ sourceEntityCode: entityCode
+ sourceEntityCodeNamespace: namespaceName [0..1]

association

associationInstance

association

A binary relation from a set of entities to a set of entities and/or data. The entityType for the class concept must be "association".

associationSource

An entity that occurs in one or more instances of a relation on the "from" (or left hand) side of a particular relation.

associationTarget

An entity on the "to" (or right hand) side of a relation.

associationData

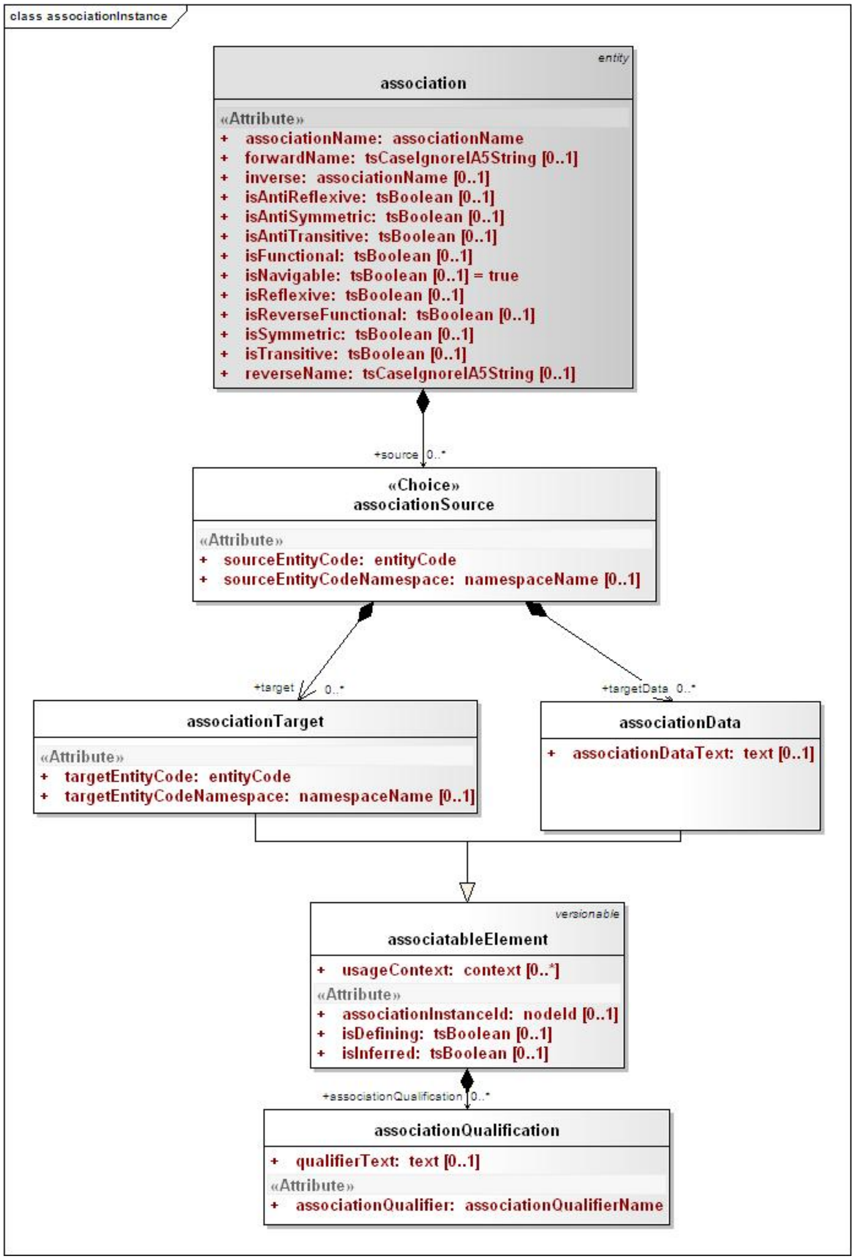
An instance of a target or RHS data value of an association.

associatableElement

Information common to both the entity and data form of the "to" (or right hand) side of an association.

associationQualification

A modifier that further qualifies the particular association instance.


class associationInstance
entity
association
«Attribute»
+ associationName: associationName
+ forwardName: tsCaseIgnoreIA5String [0..1]
+ inverse: associationName [0..1]
+ isAntiReflexive: tsBoolean [0..1]
+ isAntiSymmetric: tsBoolean [0..1]
+ isAntiTransitive: tsBoolean [0..1]
+ isFunctional: tsBoolean [0..1]
+ isNavigable: tsBoolean [0..1] = true
+ isReflexive: tsBoolean [0..1]
+ isReverseFunctional: tsBoolean [0..1]
+ isSymmetric: tsBoolean [0..1]
+ isTransitive: tsBoolean [0..1]
+ reverseName: tsCaseIgnoreIA5String [0..1]
+source 0..*
«Choice»
associationSource
«Attribute»
+ sourceEntityCode: entityCode
+ sourceEntityCodeNamespace: namespaceName [0..1]
+target 0..*
+targetData 0..*
associationTarget
«Attribute»
+ targetEntityCode: entityCode
+ targetEntityCodeNamespace: namespaceName [0..1]
associationData
+ associationDataText: text [0..1]
versionable
associatableElement
+ usageContext: context [0..*]
«Attribute»
+ associationInstanceId: nodeId [0..1]
+ isDefining: tsBoolean [0..1]
+ isInferred: tsBoolean [0..1]
+associationQualification 0..*
associationQualification
+ qualifierText: text [0..1]
«Attribute»
+ associationQualifier: associationQualifierName

associationInstance

Naming

These elements are primarily used to define metadata for a coding scheme, mapping locally used names to global references.

naming

URIMap

A local identifier that is used in a specific context (e.g. language, property name, data type, etc) and 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.

supportedAssociation

An associationName and the URI of the defining resource.

supportedAssociationQualifier

An associationQualifier and the URI of the defining resource

supportedCodingScheme

A codingSchemeName and the URI of the defining resource

supportedStatus

An entryStatus and the URI of the defining resource

supportedEntityType

An entityType and the URI of the defining resource

supportedContext

A context and the URI of the defining resource

supportedContainerName

A containerName and the URI of the defining resource

supportedDegreeOfFidelity

A degreeOfFidelity and the URI of the defining resource

supportedLanguage

A language and the URI of the defining resource

supportedProperty

A propertyName and the URI of the defining resource

supportedSortOrder

The local identifier and the URI of the defining resource

supportedHierarchy

A list of associations that can be browsed hierarchically.

supportedNamespace

A namespaceName and the corresponding URI

supportedPropertyType

A propertyType and the URI of the defining resource

supportedPropertyQualifier

A propertyQualifierName the URI of the defining resource

supportedPropertyQualifierType

A propertyQualifierType the URI of the defining resource

supportedPropertyLink

A propertyLinkName and ththe URI of the defining resource

supportedRepresentationalForm

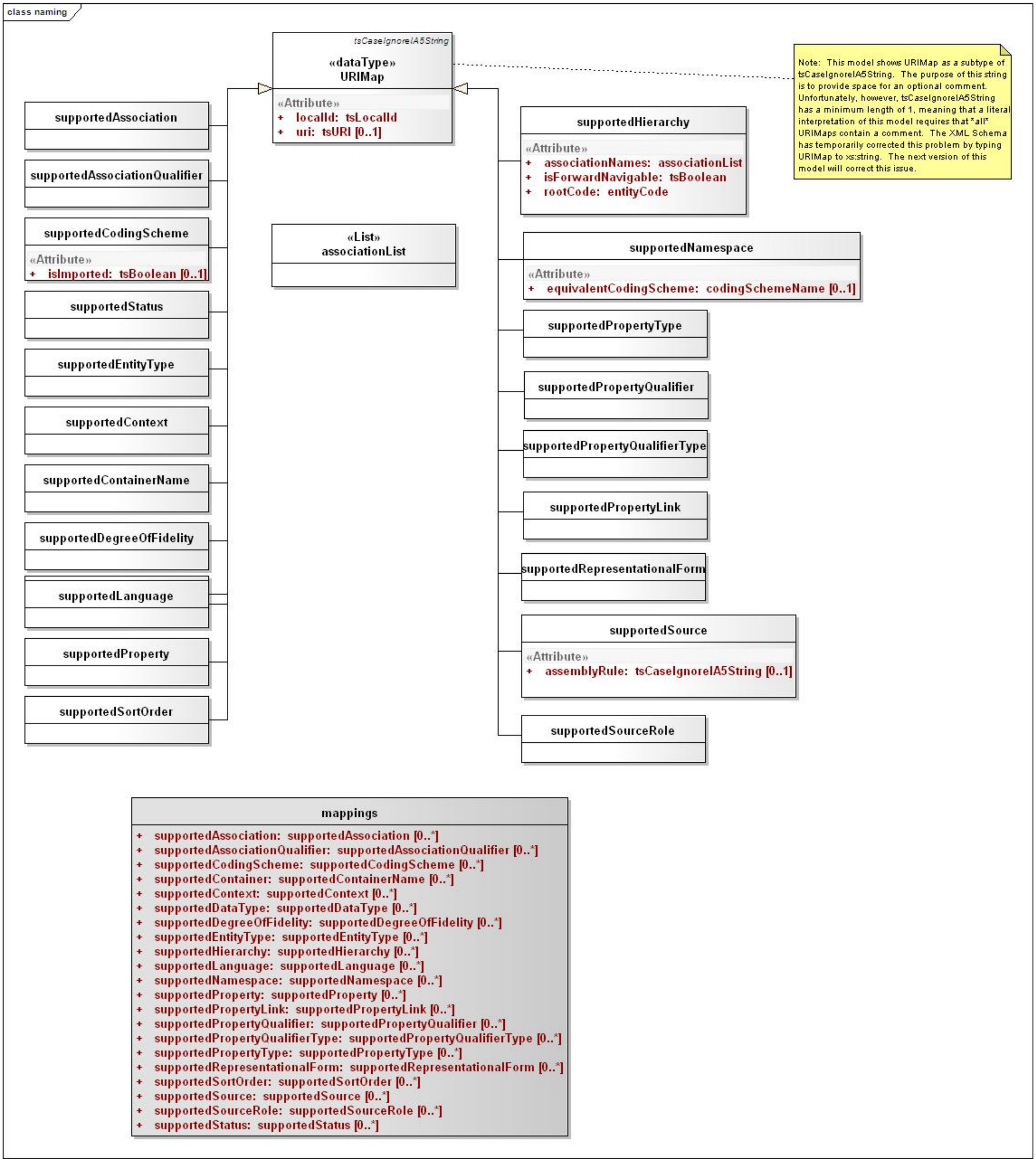
A representationalForm and the URI of the defining resource

supportedSource

A source and the URI of the defining resource. Source references can also carry an additional compositional rule section that describes how to combine a subpart such as a page number, section name, etc. with the core URI in order to form a meaningful URL. An optional role can also be specified.

supportedSourceRole

A source role and athe URI of the defining resource


class naming
tsCaseIgnoreIA5String
«dataType»
URIMap
«Attribute»
+ localId: tsLocalId
+ uri: tsURI [0..1]
Note: This model shows URIMap as a subtype of tsCaseIgnoreIA5String. The purpose of this string is to provide space for an optional comment. Unfortunately, however, tsCaseIgnoreIA5String has a minimum length of 1, meaning that a literal interpretation of this model requires that "all" URIMaps contain a comment. The XML Schema has temporarily corrected this problem by typing URIMap to xs:string. The next version of this model will correct this issue.
supportedAssociation
supportedAssociationQualifier
supportedCodingScheme
«Attribute»
+ isImported: tsBoolean [0..1]
supportedStatus
supportedEntityType
supportedContext
supportedContainerName
supportedDegreeOfFidelity
supportedLanguage
supportedProperty
supportedSortOrder
«List»
associationList
supportedHierarchy
«Attribute»
+ associationNames: associationList
+ isForwardNavigable: tsBoolean
+ rootCode: entityCode
supportedNamespace
«Attribute»
+ equivalentCodingScheme: codingSchemeName [0..1]
supportedPropertyType
supportedPropertyQualifier
supportedPropertyQualifierType
supportedPropertyLink
supportedRepresentationalForm
supportedSource
«Attribute»
+ assemblyRule: tsCaseIgnoreIA5String [0..1]
supportedSourceRole
mappings
+ supportedAssociation: supportedAssociation [0..*]
+ supportedAssociationQualifier: supportedAssociationQualifier [0..*]
+ supportedCodingScheme: supportedCodingScheme [0..*]
+ supportedContainer: supportedContainerName [0..*]
+ supportedContext: supportedContext [0..*]
+ supportedDataType: supportedDataType [0..*]
+ supportedDegreeOfFidelity: supportedDegreeOfFidelity [0..*]
+ supportedEntityType: supportedEntityType [0..*]
+ supportedHierarchy: supportedHierarchy [0..*]
+ supportedLanguage: supportedLanguage [0..*]
+ supportedNamespace: supportedNamespace [0..*]
+ supportedProperty: supportedProperty [0..*]
+ supportedPropertyLink: supportedPropertyLink [0..*]
+ supportedPropertyQualifier: supportedPropertyQualifier [0..*]
+ supportedPropertyQualifierType: supportedPropertyQualifierType [0..*]
+ supportedPropertyType: supportedPropertyType [0..*]
+ supportedRepresentationalForm: supportedRepresentationalForm [0..*]
+ supportedSortOrder: supportedSortOrder [0..*]
+ supportedSource: supportedSource [0..*]
+ supportedSourceRole: supportedSourceRole [0..*]
+ supportedStatus: supportedStatus [0..*]

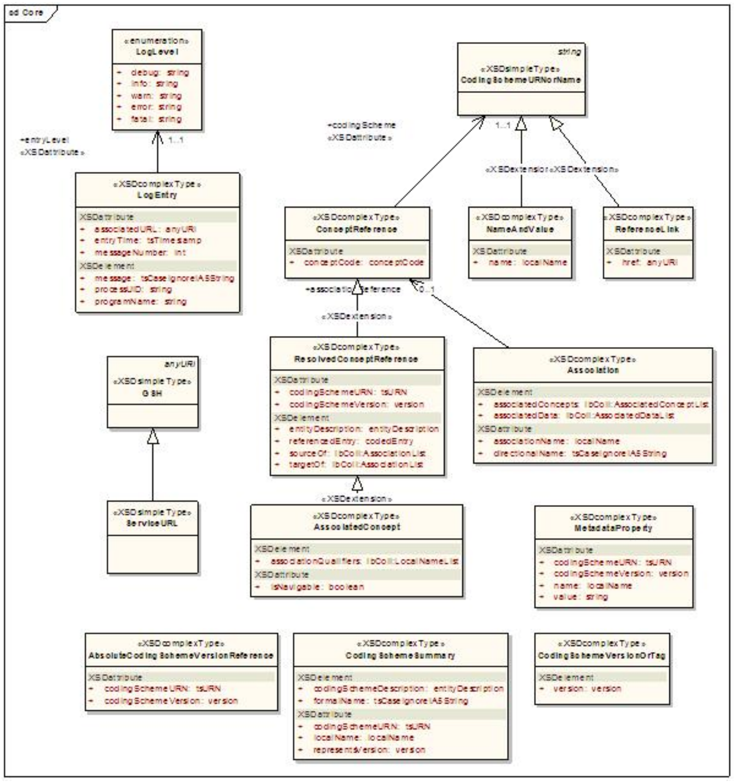
naming

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 at http://informatics.mayo.edu?page=lexex .

Core

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



Core

Components of interest include:

AbsoluteCodingSchemeVersionReference

An absolute reference to a coding scheme. This form of reference is service independent, as it doesn't depend on local coding schemes names or virtual tags.

AssociatedConcept

A concept reference that is the source or target of an association.

Association

The representation of a particular association as it appears in a CodedNode.

CodingSchemeSummary

Abbreviated list of information about a coding scheme.

CodingSchemeURNorName

Either a local name or the URN of a coding scheme. These two are differentiated syntactically - if the entity includes a colon (:) or a hash "#" it is assumed to be a URN. Otherwise it is assumed to be a local name.

CodingSchemeVersionOrTag

A named coding scheme version or a virtual tag (e.g. latest, production, etc). Note that the tagged form of identifier is only applicable in the context of a given service, as one service may identify the scheme as "production" and another as "staging".

ConceptReference

A reference to a coding scheme and a concept code.

LogEntry

A single recorded log entry.

LogLevel

Indicates severity of the log entry.

MetadataProperty

Reference to a property name and value stored in the coding scheme metadata.

NameAndValue

A simple name/value pair.

ReferenceLink

Any reference to another document element. Used by the REST architecture to embed links.

ResolvedConceptReference

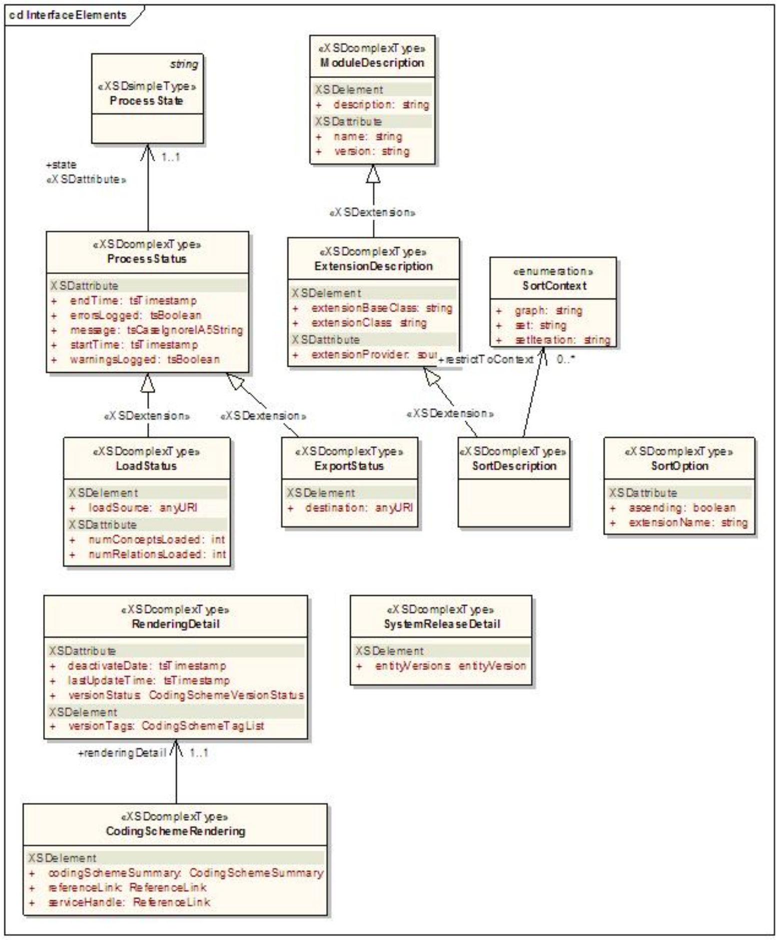
A resolvable concept reference.

ServiceURL

References a service in the Globus environment, this will be a global service handle (GSH).

InterfaceElements

Defines metadata related to model objects required by the runtime.



InterfaceElements

Components of interest include:

CodingSchemeRendering

Information about a coding scheme as it appears in a particular service.

ExportStatus

Reports the state of LexBIG export operations.

ExtensionDescription

Describes an add-on module registered to the LexBIG environment.

LoadStatus

Reports the state of LexBIG load operations.

ModuleDescription

Describes a LexBIG integrated software module.

ProcessState

Enumerates possible status reported for LexBIG runtime operations.

ProcessStatus

Reports the state of LexBIG runtime operations.

RenderingDetail

The details of how a coding scheme is rendered in a given service.

SortContext

Describes a LexBIG sort module.

SortDescription

A description of a LexBIG extension module.

SortOption

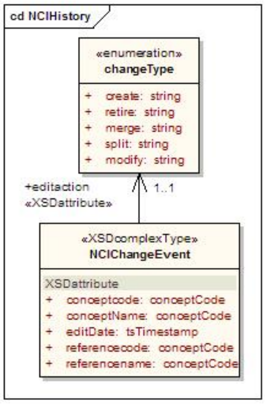
Represents a pairing of sort algorithm and order.

SystemReleaseDetail

The combination of a system release and all of the entityVersions that accompanied that release.

NCIHistory

Maintains a record of modifications made to a code system.



NCIHistory

Components of interest include:

changeType

Atomic modification actions. Currently populated from a combination of Concordia, SNOMED-CT list and NCI's action list.

NCIChangeEvent

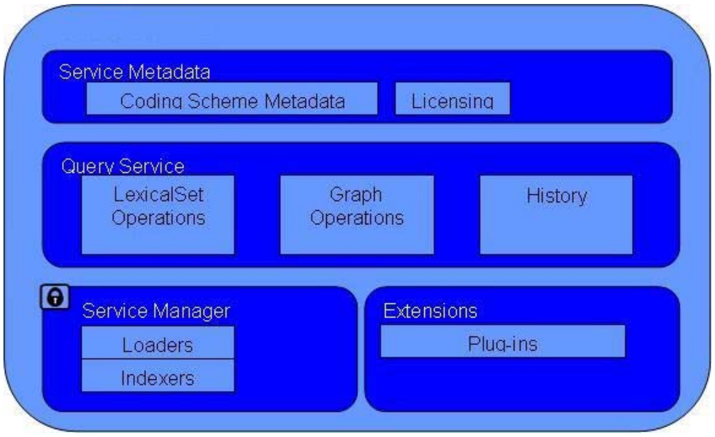
A change event as documented in ftp://ftp1.nci.nih.gov/pub/cacore/EVS/ReadMe_history.txt. Note that date and time of the change event is recorded in the containing version. All change events for the same/date and time a recorded in the same version.

Architecture

LexBIG

LexBIG Services

This section describes architectural detail for services provided by the LexBIG system. These services are geared toward the administration, management, and serving of vocabularies defined to the LexGrid/LexBIG information model. A system overview is provided, followed by a description of key subsystems and components. Each subsystem is described in terms of its overall structure, formal model, and specification of key public interfaces.

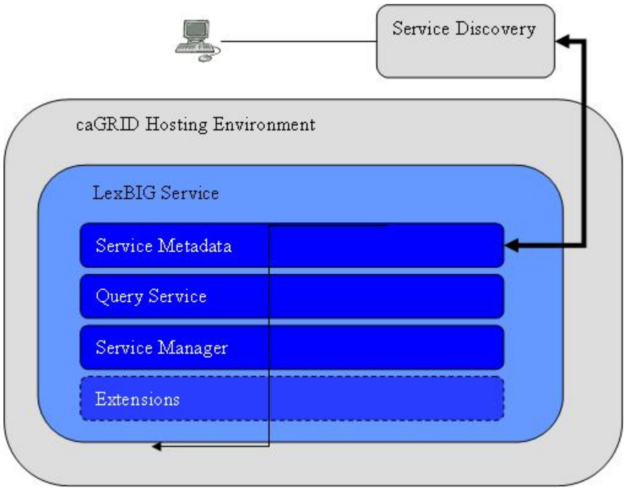


The LexBIG Service is designed to run standalone or as part of a larger network of services. It is comprised of four primary subsystems: Service Management, Service Metadata, Query Operations, and Extensions. The Service Manager provides administration control for loading a vocabulary and activating a service. The Service Metadata provides external clients with information about the vocabulary content (e.g. NCI Thesaurus) and appropriate licensing information. The Query Operations provide numerous functions for querying and traversing vocabulary content. Finally, the extensions component provides a mechanism to extend the specific service functions, such as Loaders, or re-wrap specific query operations into convenience methods. Primary points of interaction for programming include the following classes:

`LexBIGService` – This interface provides centralized access to all LexBIG services.

`LexBIGServiceManager` – The service manager provides a centralized access point for administrative functions, including write and update access for a service's content. For example, the service manager allows new coding schemes to be validated and loaded, existing coding schemes to be retired and removed, and the status of various coding schemes to be updated and changed.

caGRID Hosting

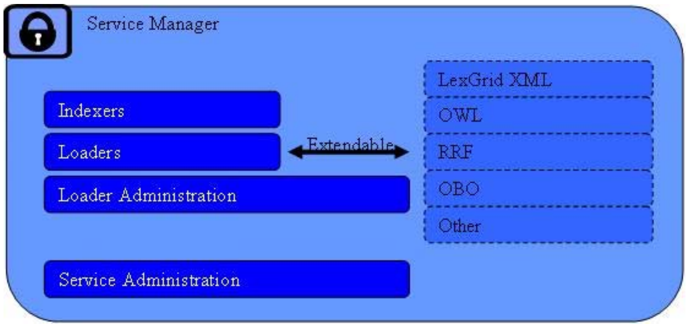


The LexBIG architecture provides the underpinnings LexBIG services to be made accessible through the caGRID environment in the future, where LexBIG services might optionally be deployed in a caGRID Globus container. caGrid provides a Globus service for service registration and discovery. LexBIG services deployed to the grid would be registered in the NCICB registry and be searchable through the NCICB index service.

Specification

Additional specifications related to the registration and discovery of LexBIG services in the caGRID environment will be included later phases of work in concordance with caGRID 1.0. This is will be coordinated with caBIG® Architecture workspace designees.

Service Management Subsystem



This subsystem provides administrative access to functions related to management and publication of LexBIG vocabularies. These functions are generally considered to be reserved for LexBIG administrators, with detailed instructions on how to secure and carry out related tasks described by the *LexBIG Administrator's Guide*.

This subsystem is further broken down into the following components:

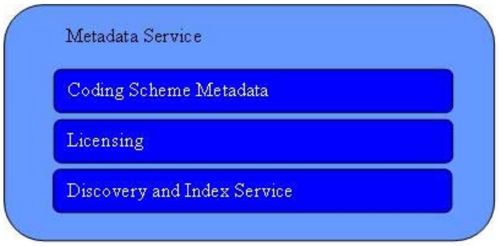
- **Indexers**
  Vocabularies may be indexed to provide enhanced performance or query capabilities. Types of indexes incorporated into the LexBIG system include but are not limited to the following:
  - Lexical Match – for example, “begins-with” and “contains”
  - Phonetic – allows for the ability to query based on “sounds-like” entry of search criteria.
  - Stemming – allows for the ability to find lexical variations of search terms.

Index creation is typically bundled into the load process. Architecturally speaking, however, this capability is decoupled and extensible.

- **Loaders**
  Vocabularies may be imported to the system from a variety of accepted formats, including but not limited to:
  - LexGrid XML (LexBIG canonical format)
  - NCI Thesaurus, provided in Web Ontology Language format (OWL)
  - UMLS Rich Release format (RRF)
  - Open Biomedical Ontologies format (OBO)

As with indexers, the load mechanism is designed to be extensible from an architectural standpoint. Additional loaders can be supported by the introduction of pluggable modules. Each module is implemented in the Java programming language according to a LexBIG-provided interface, and registered to the loader runtime environment.

Metadata and Discovery Subsystem



This subsystem provides information about accessible vocabularies, related licensing/copyright information, and registration/discovery of LexBIG services.

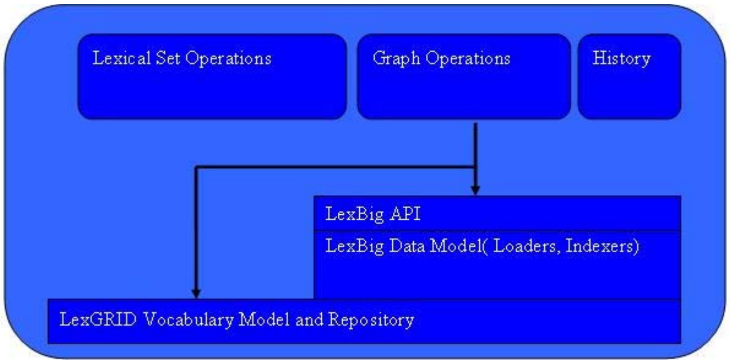
The ability to locate and resolve vocabulary metadata is fulfilled through the LexBIGService class. Metadata defined by the LexGrid information model is resolved with each CodingScheme instance. Available metadata on each resolved scheme includes, but is not necessarily limited to, the following:

- License or copyright information
- Supported values (e.g. supported concept status, language, property names, etc)
- Mappings from names used locally to globally unique URNs

In addition, each LexBIGService provides a centralized metadata index that allows registration and query of code system metadata without requiring resolution of individual CodingSchemes. This metadata index is optionally populated, typically during the vocabulary load process. The metadata index allows for the metadata of multiple code systems to be cross-indexed and searched as part of the query subsystem.

Finally, the LexBIG architecture provides the underpinnings for LexBIG services to be made accessible through the caGRID environment in the future, where vocabulary services might be deployed and discovered within a caGRID Globus container. However, this portion of the API is preliminary and awaits coordination with caBIG® Architecture WS designees to determine exact recommendations and nature of LexBIG services on the grid.

Query Subsystem


Lexical Set Operations
Graph Operations
History
LexBig API
LexBig Data Model(Loaders, Indexers)
LexGRID Vocabulary Model and Repository

This subsystem provides the functionality required to fulfill caCORE/EVS and other vocabulary requests. The Query Service is comprised of Lexical Operations, Graph Operations, Metadata, and History Operations.

Lexical Set Operations

Lexical Set Operations provides methods to return a lists or iterators of coded entries. Supported query criteria include the application of match/filter algorithms, sorting algorithms, and property restrictions. Support is also provided to resolve the union, intersection or difference of two node sets.

Graph Set Operations

Graph Operations support the subsetting of concepts according to relationship and distance, identification of relation source and target concepts, and graph traversal. Additional operations include enumeration and traversal of concepts by relation, walking of directed acyclic graphs (DAGs), enumeration of source and target concepts for a relation, and enumeration of relations for a concept.

Metadata Operations

Metadata Operations allows for the query and resolution of registered code system metadata according to specified coding scheme references, property names, or values.

History Operations

History provides vocabulary-specific information about concept insertions, modifications, splits, merges, and retirements when supplied by the content provider.

LexEVS API/Grid Service Interaction

(DESIGN DOC IMPORT START)

Revision History

Content changes to this document from the previous to the current level are indicated by revision bars (|) unless a complete rewrite is indicated.

Date	Version	Description	Author
07/29/2008	1.0	Initial document	Kevin Peterson
8/30/2008	1.1	Revised for Security and Exception Handling	Kevin Peterson
Note: If this document has been inspected, please indicate the inspection date that each version is based on in the “Change Description and Explanation” area. Entries in this log must be maintained for at least 3 years.			

Document Purpose

This document provides the detailed design and implementation of LexBIG Enterprise Vocabulary Service (LexEVS) caGrid Service. It should be noted that the LexEVS Grid Service is no longer part of the caGrid 1.1 infrastructure and will be deployed as a separate unit. This is a change from the previous release of the LexEVS Grid Service.

The LexEVS caGrid service will allow programs to utilize the caGrid 1.2 infrastructure to access LexEVS information that is currently being produced by NCICB.

Implementation Overview

Team Members

Table 1 – Team Members

Role	Name
Development Lead	Kevin Peterson
Documentation Lead	Kevin Peterson
Project Manager	Tom Johnson

Description

The LexEVS grid service will be used to obtain data accessible via the LexEVS service, specifically, the Distributed LexEVS services. Please refer to the LexEVS Programmer's Guide (https://cabig-kc.nci.nih.gov/Vocab/KC/index.php/LexEVS_5.0_Programmer%27s_Guide) for more information.

For more Documentation, Build/Deployment instructions and examples, visit the project documentation home at: http://gforge.nci.nih.gov/docman/index.php?group_id=491&selected_doc_group_id=3749&language_id=1

Scope

The LexEVS Grid service will provide programmatic access to the LexBIG domain objects that are available via the LexBIG information model.

The LexEVS grid service will be registered in Cancer Data Standards Repository (caDSR) under the following category:

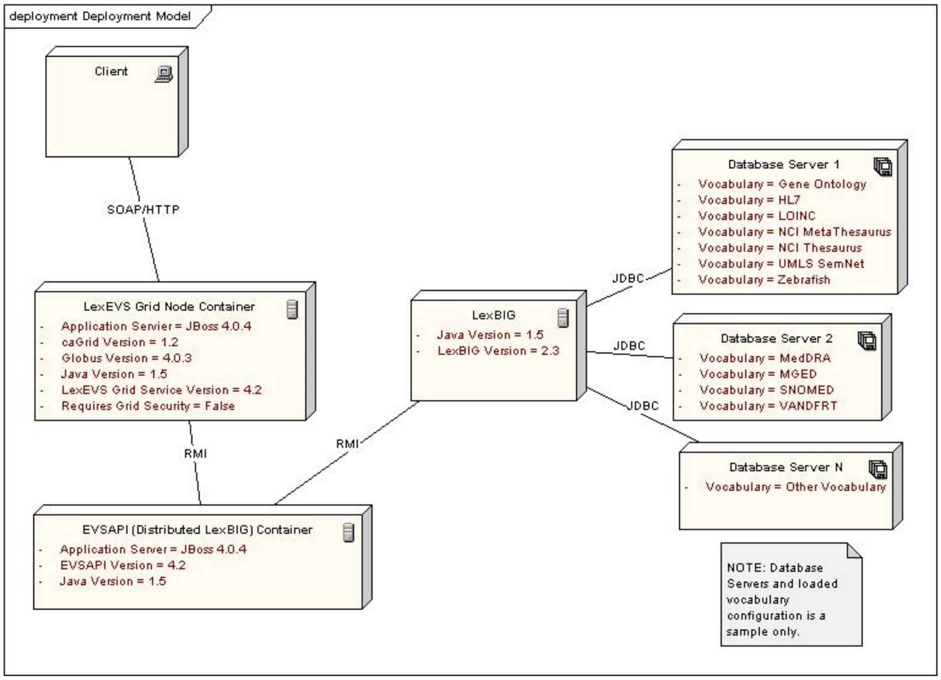
LexEVS Grid Service	
Context	caBIG
Classification Scheme	LexBIG
Version	LexBIG_v2_3_rv1

Architecture

The LexEVS Grid Service is implemented to expose the API and Model of LexBIG 2.3. For more information on LexBIG, see http://informatics.mayo.edu

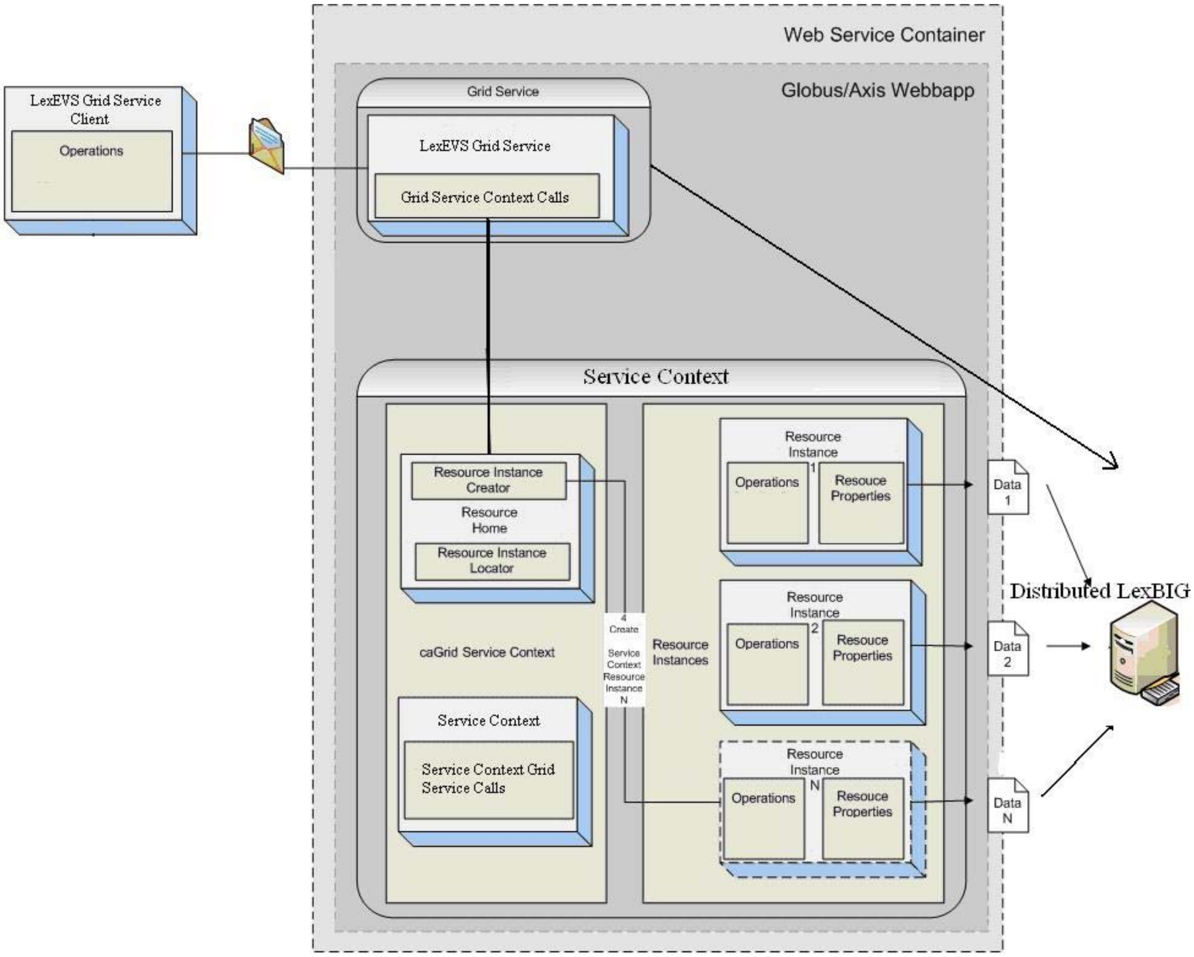
LexEVS Grid Service is deployed in a JBoss (http://www.jboss.org/) Application Server, inside of a Globus (http://www.globus.org/) Web Application installation. LexEVS Grid Service depends on LexEVS API(http://lexevsapi.nci.nih.gov/), which is also deployed to a JBoss container. For more information on the deployment of EVSAPI, see http://gforge.nci.nih.gov/docman/index.php?group_id=366&selected_doc_group_id=1914&language_id=1 LexEVS API itself depends on an installation of LexBIG (http://informatics.mayo.edu).

The diagram below shows the various components of the LexEVS Grid Service System and how they interact.


deployment Deployment Model
Client
SOAP/HTTP
LexEVS Grid Node Container
- Application Servier = JBoss 4.0.4
- caGrid Version = 1.2
- Globus Version = 4.0.3
- Java Version = 1.5
- LexEVS Grid Service Version = 4.2
- Requires Grid Security = False
RMI
RMI
EVSAPI (Distributed LexBIG) Container
- Application Server = JBoss 4.0.4
- EVSAPI Version = 4.2
- Java Version = 1.5
LexBIG
- Java Version = 1.5
- LexBIG Version = 2.3
JDBC
JDBC
JDBC
Database Server 1
- Vocabulary = Gene Ontology
- Vocabulary = HL7
- Vocabulary = LOINC
- Vocabulary = NCI MetaThesaurus
- Vocabulary = NCI Thesaurus
- Vocabulary = UMLS SemNet
- Vocabulary = Zebrafish
Database Server 2
- Vocabulary = MedDRA
- Vocabulary = MGED
- Vocabulary = SNOMED
- Vocabulary = VANDFRT
Database Server N
- Vocabulary = Other Vocabulary
NOTE: Database Servers and loaded vocabulary configuration is a sample only.

LexEVS Grid Service and EVSAPI need not be deployed to physically separate servers, but it is recommended that if they are co-located on the same server, they should be deployed to separate JBoss containers.

Below is the LexEVS Grid Service Architecture, viewed from inside of the Web Service Container. For more information on how Service Contexts and Resources are used, see the “Service Contexts and State” section below.



LexEVS Grid Service Class Diagram

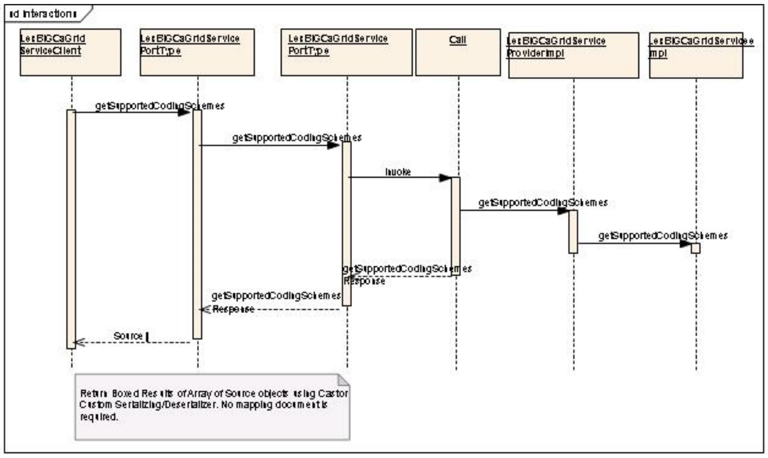
The LexEVS Grid Service is built on the LexGrid/LexBIG model and implementation. For more information about this model, visit (LexBIG) https://gforge.nci.nih.gov/plugins/scmsvn/viewcvs.php/LexBIG_Core_Services/LexBIG-2.3/lexbig/lbModel/?root=lexevs and (LexGrid) https://gforge.nci.nih.gov/plugins/scmsvn/viewcvs.php/LexBIG_Core_Services/LexBIG-2.3/lgModel/?root=lexevs

Also, visit http://informatics.mayo.edu for background information as well as Class Diagrams, examples, and other information.

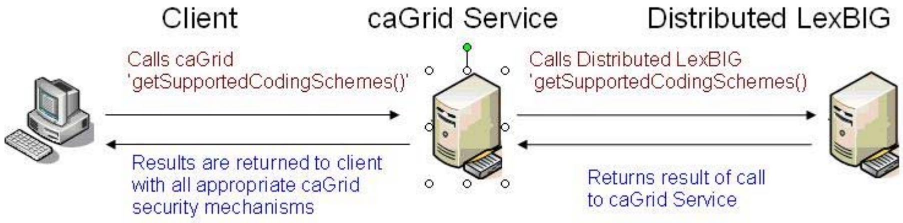
For information specific to the LexEVS Grid Service, visit: https://gforge.nci.nih.gov/plugins/scmsvn/viewcvs.php/LexBIG_Core_Services/LexBIG-2.3/lexbig/lbModel.cagrid/?root=lexevs This link contains Class Diagrams and descriptions for input/output parameters, as well as other information concerning the Silver Level Compliance submission package.

LexEVS Grid Service Sequence Diagram

The sequence diagram for the operation “getSupportedCodingSchemes” is described below:


sd Interactions
LexBIGCaGrid ServiceClient
LexBIGCaGridService PortType
LexBIGCaGridService PortType
Call
LexBIGCaGridService ProviderImpl
LexBIGCaGridService Impl
getSupportedCodingSchemes
getSupportedCodingSchemes
invoke
getSupportedCodingSchemes
getSupportedCodingSchemes
getSupportedCodingSchemes Response
getSupportedCodingSchemes Response
Source[]
Return Boxed Results of Array of Source objects using Castor Custom Serializing/Deserializer. No mapping document is required.

General Call Sequence Example:



Assumptions

- The LexEVS service will be based on the latest LexEVS 5.0 release.
- The LexEVS Grid Service will not have any method level security. All security requirements will be handled by the actual deployment of the underlying LexEVS 5.0 service. Please see the "Security" section below for more information on how the LexEVS Grid Service utilizes this security.
- The LexEVS Grid Service will not be deployed as a "core" service by caGrid at NCICB as was previously done, but rather will now be deployed as a standalone service.
- The LexEVS Grid Service release schedule will no longer be coupled to the caGrid deployment schedule as previously done.
- Multiple version of LexEVS Grid Service may be active at the same instance in time depending solely on the availability of the underlining EVSAPI service.

Dependencies

- LexEVS 5.0 service needs to be available and running correctly.
- The LexEVS service and operations will use the Introduce toolkit to generate the appropriate structure for registering the service into caDSR.

Third Party Tools

- Introduce Toolkit
- Globus Toolkit (4.0.3) or appropriate version supported by caGrid 1.2
- caGrid 1.2 core infrastructure

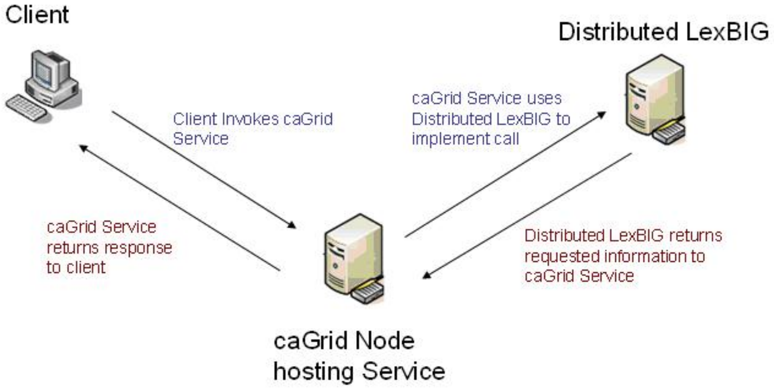
Server

The LexEVS Grid Service will be deployed as a "stand alone" grid service at NCICB.

APIs

The main Service API exposed by the LexEVS Grid service will be the http://informatics.mayo.edu/LexGrid/downloads/javadocGrid/org/LexGrid/LexBIG/cagrid/interfaces/LexBIGServiceGrid.html Interface. All other APIs will not be directly exposed, but will be made available through Service Contexts.

In General, API calls will follow this sequence:



API Examples

For an example clients, service calls, and SOAP messages, see http://gforge.nci.nih.gov/docman/index.php?group_id=491&selected_doc_group_id=3880&language_id=1

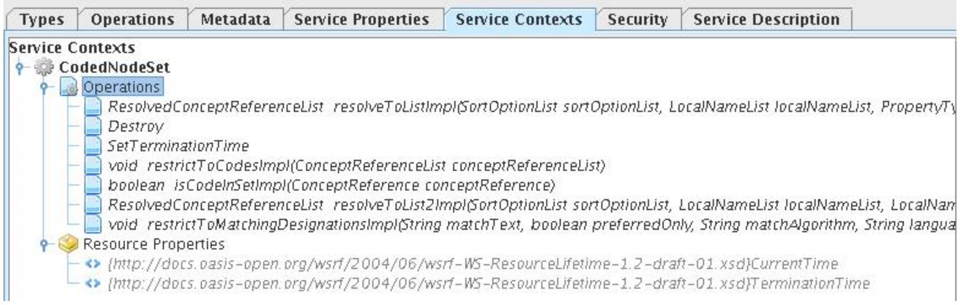
Example API usage:

Searching for concepts in NCI Thesaurus containing the string "Gene"

```
//Create a Connection to the Grid Service
LexBIGServiceGrid lbs = new LexBIGServiceGridAdapter
(gridServiceURL);

//Set up the CodingSchemeIdentification object to define the
Coding Scheme</font>
CodingSchemeIdentification csid = new
CodingSchemeIdentification();
csid.setName("NCI Thesaurus");

//Get the CodedNodeSet for that CodingScheme (This returns a
CodedNodeSet Service Context)
CodedNodeSetGrid cnsg = lbs.getCodingSchemeConcepts(csid,
null);
//getCodingSchemeConcepts is a Grid Service Call

//Set the text to match
MatchCriteria matchText = new MatchCriteria();
matchText.setText("Gene");
//Define a SearchDesignationOption, if any
SearchDesignationOption searchOption = new
SearchDesignationOption();

//Choose an algorithm to do the matching
ExtensionIdentification matchAlgorithm = new
ExtensionIdentification();
matchAlgorithm.setLexBIGExtensionName("contains");

//Chose a language
LanguageIdentification language = new
LanguageIdentification();
language.setIdentifier("en");

//Restrict the CodedNodeSet
cnsg.restrictToMatchingDesignations(matchText,
searchOption, matchAlgorithm, language);
//restrictToMatchingDesignations is a Grid Service Call

//Create a SetResolutionPolicy to handle the details of
Resolving the CodedNodeSet
//Here, we will set the Maximum number of Concepts
returned to 10.
SetResolutionPolicy resolvePolicy = new
SetResolutionPolicy();
resolvePolicy.setMaximumToReturn(10);

//Do the resolve
ResolvedConceptReferenceList rcrlist = cnsg.resolveToList(resolvePolicy);
//resolveToList is a Grid Service Call

//Use the returned ResolvedConceptReferenceList to
print some details about the concepts found
ResolvedConceptReference[] rcref = rcrlist.getResolvedConceptReference();
for (int i = 0; i < rcref.length; i++) {
        System.out.println(rcref[i].getConceptCode());
        System.out.println(rcref[i].getReferencedEntry().
                getPresentation()[0].getText().getContent());
}
```

Service Contexts and State

Along with the Main Service (described above), the Server will also host the following Service Contexts. These Service Contexts are not meant to be called directly as Grid Services. The main function of these Service Contexts is to provide additional functionality to the Main Service.



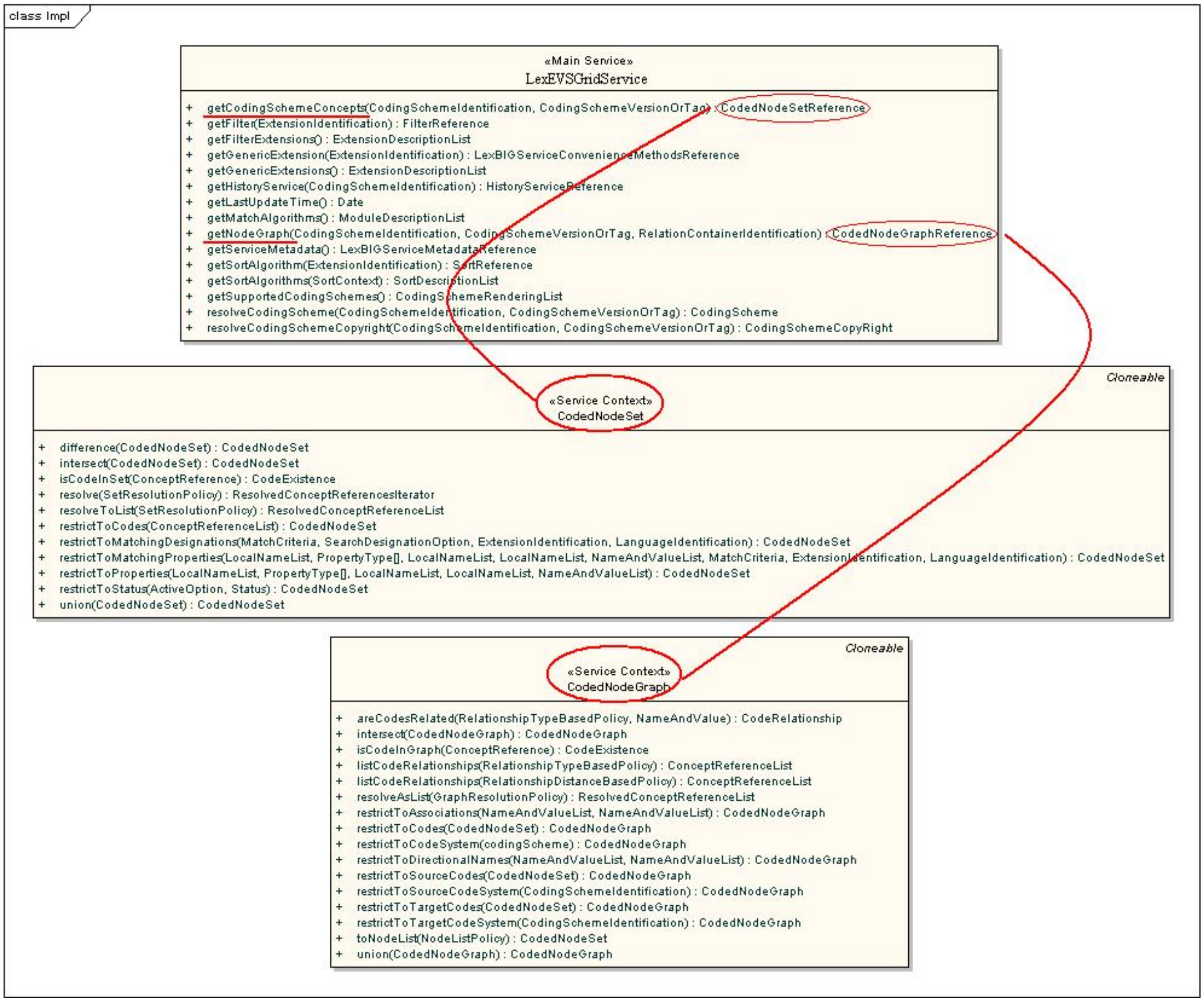
```
Service Contexts
  CodedNodeSet
    Operations
      ResolvedConceptReferenceList  resolveToListImpl(SortOptionList sortOptionList, LocalNameList localNameList, PropertyTy
      Destroy
      SetTerminationTime
      void  restrictToCodesImpl(ConceptReferenceList conceptReferenceList)
      boolean  isCodeInSetImpl(ConceptReference conceptReference)
      ResolvedConceptReferenceList  resolveToList2Impl(SortOptionList sortOptionList, LocalNameList localNameList, LocalNam
      void  restrictToMatchingDesignationsImpl(String matchText, boolean preferredOnly, String matchAlgorithm, String langua
    Resource Properties
      <> {http://docs.oasis-open.org/wsrf/2004/06/wsrf-WS-ResourceLifetime-1.2-draft-01.xsd}CurrentTime
      <> {http://docs.oasis-open.org/wsrf/2004/06/wsrf-WS-ResourceLifetime-1.2-draft-01.xsd}TerminationTime
```

Service Context Operations Example in Introduce

IMPORTANT: Service Contexts are only meant to be called through the Main Service – not directly. Through the Main Service, References to these Service Contexts can be obtained. Calls are made to the Service Contexts through these References.

Obtaining a Service Context Reference

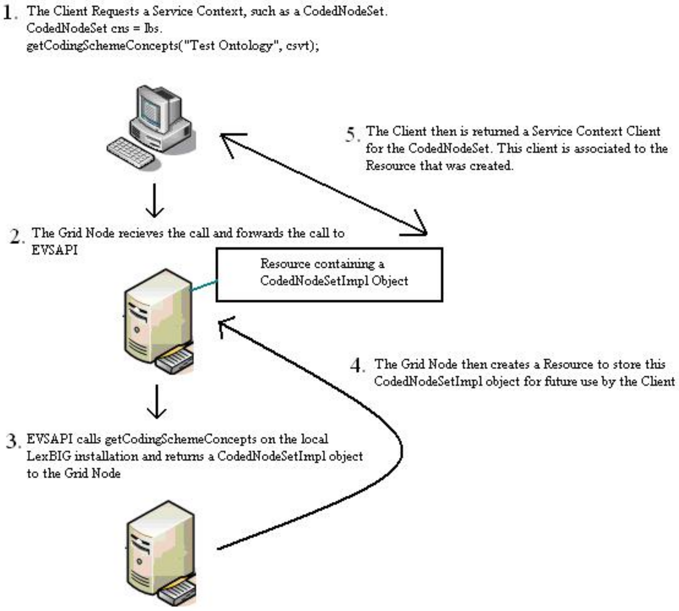
In the figure below, two LexEVS Grid Service Calls are highlighted, ‘getCodingSchemeConcepts’ and ‘getNodeGraph’. These two Grid Service Calls have been selected because they return to the user a “Reference” to a Service Context. For ‘getCodingSchemeConcepts’, the return type is CodedNodeSetReference (which references the CodedNodeSet Service Context). For ‘getNodeGraph’, the return type is CodedNodeGraphReference (which references the CodedNodeGraph Service Context).


class Impl
«Main Service»
LexEVSGridService
+ getCodingSchemeConcepts(CodingSchemeIdentification, CodingSchemeVersionOrTag) : CodedNodeSetReference
+ getFilter(ExtensionIdentification) : FilterReference
+ getFilterExtensions() : ExtensionDescriptionList
+ getGenericExtension(ExtensionIdentification) : LexBIGServiceConvenienceMethodsReference
+ getGenericExtensions() : ExtensionDescriptionList
+ getHistoryService(CodingSchemeIdentification) : HistoryServiceReference
+ getLastUpdateTime() : Date
+ getMatchAlgorithms() : ModuleDescriptionList
+ getNodeGraph(CodingSchemeIdentification, CodingSchemeVersionOrTag, RelationContainerIdentification) : CodedNodeGraphReference
+ getServiceMetadata() : LexBIGServiceMetadataReference
+ getSortAlgorithm(ExtensionIdentification) : SortReference
+ getSortAlgorithms(SortContext) : SortDescriptionList
+ getSupportedCodingSchemes() : CodingSchemeRenderingList
+ resolveCodingScheme(CodingSchemeIdentification, CodingSchemeVersionOrTag) : CodingScheme
+ resolveCodingSchemeCopyright(CodingSchemeIdentification, CodingSchemeVersionOrTag) : CodingSchemeCopyRight
Cloneable
«Service Context»
CodedNodeSet
+ difference(CodedNodeSet) : CodedNodeSet
+ intersect(CodedNodeSet) : CodedNodeSet
+ isCodeInSet(ConceptReference) : CodeExistence
+ resolve(SetResolutionPolicy) : ResolvedConceptReferencesIterator
+ resolveToList(SetResolutionPolicy) : ResolvedConceptReferenceList
+ restrictToCodes(ConceptReferenceList) : CodedNodeSet
+ restrictToMatchingDesignations(MatchCriteria, SearchDesignationOption, ExtensionIdentification, LanguageIdentification) : CodedNodeSet
+ restrictToMatchingProperties(LocalNameList, PropertyType[], LocalNameList, LocalNameList, NameAndValueList, MatchCriteria, ExtensionIdentification, LanguageIdentification) : CodedNodeSet
+ restrictToProperties(LocalNameList, PropertyType[], LocalNameList, LocalNameList, NameAndValueList) : CodedNodeSet
+ restrictToStatus(ActiveOption, Status) : CodedNodeSet
+ union(CodedNodeSet) : CodedNodeSet
Cloneable
«Service Context»
CodedNodeGraph
+ areCodesRelated(RelationshipTypeBasedPolicy, NameAndValue) : CodeRelationship
+ intersect(CodedNodeGraph) : CodedNodeGraph
+ isCodeInGraph(ConceptReference) : CodeExistence
+ listCodeRelationships(RelationshipTypeBasedPolicy) : ConceptReferenceList
+ listCodeRelationships(RelationshipDistanceBasedPolicy) : ConceptReferenceList
+ resolveAsList(GraphResolutionPolicy) : ResolvedConceptReferenceList
+ restrictToAssociations(NameAndValueList, NameAndValueList) : CodedNodeGraph
+ restrictToCodes(CodedNodeSet) : CodedNodeGraph
+ restrictToCodeSystem(codingScheme) : CodedNodeGraph
+ restrictToDirectionalNames(NameAndValueList, NameAndValueList) : CodedNodeGraph
+ restrictToSourceCodes(CodedNodeSet) : CodedNodeGraph
+ restrictToSourceCodeSystem(CodingSchemeIdentification) : CodedNodeGraph
+ restrictToTargetCodes(CodedNodeSet) : CodedNodeGraph
+ restrictToTargetCodeSystem(CodingSchemeIdentification) : CodedNodeGraph
+ toNodeList(NodeListPolicy) : CodedNodeSet
+ union(CodedNodeGraph) : CodedNodeGraph

Resources

LexEVS Grid Services use the WS-Resource Framework (WSRF) to allow for stateful calls to the server. When a client requests a Service Context, the client is not only issued a Reference to the Service Context that was requested, but to a unique stateful Resource on the server as well. This Resource is used in the LexEVS Grid Services as a way of statefully holding objects for further use by the client. For more information about how caGrid uses the WS-Resource Framework (WSRF), see http://www.cagrid.org/wiki/Metadata:WSRF For more information on how Resources are implemented in the LexEVS Grid Service, see http://gforge.nci.nih.gov/docman/view.php/491/13736/LexEVSGrid.ppt

Service Context Sequence Service Contexts API calls follow this general process:


1. The Client Requests a Service Context, such as a CodedNodeSet.
CodedNodeSet cns = lbs.
getCodingSchemeConcepts("Test Ontology", csvt);

5. The Client then is returned a Service Context Client for the CodedNodeSet. This client is associated to the Resource that was created.

2. The Grid Node recieves the call and forwards the call to EVSAPI

Resource containing a CodedNodeSetImpl Object

4. The Grid Node then creates a Resource to store this CodedNodeSetImpl object for future use by the Client

3. EVSAPI calls getCodingSchemeConcepts on the local LexBIG installation and returns a CodedNodeSetImpl object to the Grid Node

Service Context and Resource Assignment

NOTE: By default, these services are destroyed 5 minutes after creation.

Below is a listing of the supported Service Contexts:

1. CodedNodeSet

http://informatics.mayo.edu/LexGrid/downloads/javadocGrid/org/LexGrid/LexBIG/cagrid/interfaces/CodedNodeSetGrid.html

To construct a CodedNodeSet, the user calls getCodingSchemeConcepts as described above. When the user creates a CodedNodeSet through the API call getCodingSchemeConcepts, the server creates and stores the CodedNodeSet server-side as a Resource. This Resource is associated with the client and will be accessible only by the client that created it.

CodedNodeSet Call Sequence:

1. The user requests a CodedNodeSet using getCodingSchemeConcepts.

```
	CodedNodeSetGrid cns = lbs.getCodingSchemeConcepts(
	org.LexGrid.LexBIG.DataModel.cagrid.CodingSchemeIdentification,
	org.LexGrid.LexBIG.DataModel.Core.CodingSchemeVersionOrTag);
```

2. The server calls the Distributed LexBIG getCodingSchemeConcepts method, returning to the server an org.LexGrid.LexBIG.Impl.CodedNodeSetImpl (the implementation of org.LexGrid.LexBIG.LexBIGService.CodedNodeSet) object.

3. The server then creates an org.LexGrid.LexBIG.cagrid.LexBIGCaGridServices.CodedNodeSet.service.globus.resource.CodedNodeSetResource. This Resource will be used to hold the instance of org.LexGrid.LexBIG.Impl.CodedNodeSetImpl, the implementation of org.LexGrid.LexBIG.LexBIGService.CodedNodeSet that was created above.

4. The server returns an org.LexGrid.LexBIG.cagrid.LexBIGCaGridServices.CodedNodeSet.stubs.types.CodedNodeSetReference object to the client. This is the reference to the CodedNodeSet Service Context. This object has a direct reference to the Resource created above. The user now uses this client to make transparent Grid calls through the Service Context.

5. The client may continue to make statefull calls to the CodedNodeSetClient and the assigned Resource.

6. These restrictions are separate calls but statefully maintained on the server via the Resource.

2. CodedNodeGraph

http://informatics.mayo.edu/LexGrid/downloads/javadocGrid/org/LexGrid/LexBIG/cagrid/interfaces/CodedNodeGraphGrid.html

To construct a CodedNodeGraph, the user calls getNodeGraph as described above. When the user creates a CodedNodeGraph through the API call getNodeGraph, the server creates and stores the CodedNodeGraph server-side as a Resource. This Resource is associated with the client and will be accessible only by the client that created it.

CodedNodeGraph Call Sequence:

1. The user requests a CodedNodeGraph using getCodingSchemeConcepts.

```
	CodedNodeGraphGrid cng = client.getNodeGraph(
	org.LexGrid.LexBIG.DataModel.cagrid.CodingSchemeIdentification,
	org.LexGrid.LexBIG.DataModel.Core.CodingSchemeVersionOrTag,
	org.LexGrid.LexBIG.DataModel.cagrid.
	RelationContainerIdentification);
```

2. The server calls the Distributed LexBIG getNodeGraph method, returning to the server an org.LexGrid.LexBIG.Impl.CodedNodeGraphImpl (the implementation of org.LexGrid.LexBIG.LexBIGService.CodedNodeGraph) object.

3. The server then creates an org.LexGrid.LexBIG.cagrid.LexBIGCaGridServices.CodedNodeGraph.service.globus.resource.CodedNodeGraphResource. This Resource will be used to hold the instance of org.LexGrid.LexBIG.Impl.CodedNodeGraphImpl, the implementation of org.LexGrid.LexBIG.LexBIGService.CodedNodeGraph that was created above.

4. The server returns an org.LexGrid.LexBIG.cagrid.LexBIGCaGridServices.CodedNodeGraph.stubs.types.CodedNodeGraphReference object to the client. This is the reference to the CodedNodeGraph Service Context. This object has a direct reference to the Resource created above. The user now uses this client to make transparent Grid calls through the Service Context.

5. The client may continue to make statefull calls to the CodedNodeGraphClient and the assigned Resource. For example, the client may add Restrictions to the CodedNodeGraph before a Resolve:

```
cng.restrictToCodeSystem(org.LexGrid.LexBIG.DataModel.cagrid.
	CodingSchemeIdentification);
```

6. These restrictions are separate calls but statefully maintained on the server via the Resource.

3. LexBIGServiceConvenienceMethods

http://informatics.mayo.edu/LexGrid/downloads/javadocGrid/org/LexGrid/LexBIG/cagrid/interfaces/LexBIGServiceConvenienceMethodsGrid.html

To construct a LexBIGServiceConvenienceMethods, the user calls getGenericExtensions as described above. When the user creates a LexBIGServiceConvenienceMethods through the API call getGenericExtensions, the server creates and stores the LexBIGServiceConvenienceMethods server-side as a Resource. This Resource is associated with the client and will be accessible only by the client that created it.

LexBIGServiceConvenienceMethods Call Sequence:

1. The user requests a LexBIGServiceConvenienceMethods using getGenericExtensions.

```
	LexBIGServiceConvenienceMethodsGrid lbscm = 					lbs.getGenericExtensions(
		org.LexGrid.LexBIG.DataModel.cagrid.ExtensionIdentification);
```

2. The server calls the Distributed LexBIG getGenericExtensions method, returning to the server an org.LexGrid.LexBIG.Impl.Extensions.GenericExtensions.LexBIGServiceConvenienceMethodsImpl (the implementation of org.LexGrid.LexBIG.Extensions.Generic.LexBIGServiceConvenienceMethods) object.

3. The server then creates an org.LexGrid.LexBIG.cagrid.LexBIGCaGridServices.LexBIGServiceConvenienceMethods.service.globus.resource.LexBIGServiceConvenienceMethodsResource. This Resource will be used to hold the instance of org.LexGrid.LexBIG.Impl.Extensions.GenericExtensions.LexBIGServiceConvenienceMethodsImpl, the implementation of org.LexGrid.LexBIG.Extensions.Generic.LexBIGServiceConvenienceMethods that was created above.

4. The server returns an org.LexGrid.LexBIG.cagrid.LexBIGCaGridServicesLexBIGServiceConvenienceMethods.stubs.types.LexBIGServiceConvenienceMethodsReference object to the client. This is the reference to the LexBIGServiceConvenienceMethods Service Context. This object has a direct reference to the Resource created above. This LexBIGServiceConvenienceMethodsClient implements org.LexGrid.LexBIG.Extensions.Generic.LexBIGServiceConvenienceMethods. The user now uses this client to make transparent Grid calls through the Service Context. Because this LexBIGServiceConvenienceMethods implements org.LexGrid.LexBIG.Extensions.Generic.LexBIGServiceConvenienceMethods, API calls will look to the user as being identical to direct LexBIG API calls.

5. The client may continue to make statefull calls to the LexBIGServiceConvenienceMethods Client and the assigned Resource.

6. These API calls are separate calls but statefully maintained on the server via the Resource.

4. LexBIGServiceMetadata

http://informatics.mayo.edu/LexGrid/downloads/javadocGrid/org/LexGrid/LexBIG/cagrid/interfaces/LexBIGServiceMetadataGrid.html

To construct a LexBIGServiceMetadata, the user calls getServiceMetadata as described above. When the user creates a LexBIGServiceMetadata through the API call getServiceMetadata , the server creates and stores the LexBIGServiceMetadata server-side as a Resource. This Resource is associated with the client and will be accessible only by the client that created it.

LexBIGServiceMetadata Call Sequence:

1. The user requests a LexBIGServiceMetadata using getServiceMetadata.

```
LexBIGServiceMetadataGrid metadata = lbs.getServiceMetadata();
```

2. The server calls the Distributed LexBIG getServiceMetadata method, returning to the server an implementation of org.LexGrid.LexBIG.LexBIGService.LexBIGServiceMetadata object.

3. The server then creates an org.LexGrid.LexBIG.cagrid.LexBIGCaGridServices.LexBIGServiceMetadata.service.globus.resource.LexBIGServiceMetadataResource. This Resource will be used to hold the instance of an implementation of org.LexGrid.LexBIG.LexBIGService.LexBIGServiceMetadata.

4. org.LexGrid.LexBIG.cagrid.LexBIGCaGridServices.LexBIGServiceMetadata.stubs.types.LexBIGServiceMetadata object to the client. This is the reference to the LexBIGServiceMetadata Service Context. This object has a direct reference to the Resource created above. The user now uses this client to make transparent Grid calls through the Service Context.

5. The client may continue to make statefull calls to the LexBIGServiceMetadata and the assigned Resource.

6. These API calls are separate calls but statefully maintained on the server via the Resource.

5. HistoryService

http://informatics.mayo.edu/LexGrid/downloads/javadocGrid/org/LexGrid/LexBIG/cagrid/interfaces/HistoryServiceGrid.html

To construct a HistoryService, the user calls getHistoryService as described above. When the user creates a HistoryService through the API call getHistoryService, the server creates and stores the HistoryService server-side as a Resource. This Resource is associated with the client and will be accessible only by the client that created it.

HistoryService Call Sequence:

1. The user requests a HistoryService using getHistoryService .

```
HistoryServiceGrid history = lbs.getHistoryService(
        org.LexGrid.LexBIG.DataModel.cagrid.CodingSchemeIdentification);
```

2. The server calls the Distributed LexBIG getHistoryService method, returning to the server an implementation of org.LexGrid.LexBIG.History.HistoryService object.

3. The server then creates an org.LexGrid.LexBIG.cagrid.LexBIGCaGridServices.HistoryService.service.globus.resource.HistoryServiceResource. This Resource will be used to hold the instance of an implementation of org.LexGrid.LexBIG.History.HistoryService.

4. The server returns an org.LexGrid.LexBIG.cagrid.LexBIGCaGridServices.LexBIGServiceMetadata.stubs.types.LexBIGServiceMetadata object to the client. This is the reference to the HistoryService Service Context. This object has a direct reference to the Resource created above. The user now uses this client to make transparent Grid calls through the Service Context.

5. The client may continue to make statefull calls to the HistoryServiceClient and the assigned Resource. For example, the client may call any method in org.LexGrid.LexBIG.History.HistoryService

Example: history.getLatestBaseline();

6. These API calls are separate calls but statefully maintained on the server via the Resource.

6. Sort

http://informatics.mayo.edu/LexGrid/downloads/javadoc/org/LexGrid/LexBIG/Extensions/Query/Sort.html

To construct a Sort, the user calls getSortAlgorithm as described above. When the user creates a Sort through the API call getSortAlgorithm, the server creates and stores the Sort server-side as a Resource. This Resource is associated with the client and will be accessible only by the client that created it.

Sort Call Sequence:

1. The user requests a Sort using getSortAlgorithm .

```
Sort sort = lbs.getSortAlgorithm(
        org.LexGrid.LexBIG.DataModel.cagrid.
        ExtensionIdentification);
```

2. The server calls the Distributed LexBIG getSortAlgorithm method, returning to the server an implementation of org.LexGrid.LexBIG.Extensions.Query.Sort) object.

3. The server then creates an org.LexGrid.LexBIG.cagrid.LexBIGCaGridServices.Sort .service.globus.resource.Sort Resource. This Resource will be used to hold the instance of an implementation of org.LexGrid.LexBIG.Extensions.Query.Sort.

4. The server returns an org.LexGrid.LexBIG.cagrid.LexBIGCaGridServices.service.SortClient object to the client. This is the client to the Sort Service Context. This object has a direct reference to the Resource created above. This SortClient implements org.LexGrid.LexBIG.Extensions.Query.Sort. The user now uses this client to make transparent Grid calls through the Service Context. Because this Sort implements org.LexGrid.LexBIG.Extensions.Query.Sort, API calls will look to the user as being identical to direct LexBIG API calls.

5. The client may continue to make statefull calls to the SortClient and the assigned Resource. For example, the client may call any method in org.LexGrid.LexBIG.Extensions.Query.Sort

```
sort.compare(codedNodeReference1, codedNodeReference2);
```

6. These API calls are separate calls but statefully maintained on the server via the Resource.

7. Filter

http://informatics.mayo.edu/LexGrid/downloads/javadoc/org/LexGrid/LexBIG/Extensions/Query/Filter.htm

To construct a Filter, the user calls getFilter as described above. When the user creates a Filter through the API call getFilter, the server creates and stores the Sort server-side as a Resource. This Resource is associated with the client and will be accessible only by the client that created it.

Filter Call Sequence:

1. The user requests a Filter using getFilter

```
Filter filter = lbs.getFilter(org.LexGrid.LexBIG.DataModel.cagrid.
        ExtensionIdentification);
```

2. The server calls the Distributed LexBIG getFilter method, returning to the server an implementation of org.LexGrid.LexBIG.Extensions.Query.Filter) object.

3. The server then creates an org.LexGrid.LexBIG.cagrid.LexBIGCaGridServices.Filter.service.globus.resource.FilterResource. This Resource will be used to hold the instance of an implementation of org.LexGrid.LexBIG.Extensions.Query.Filter.

4. The server returns an org.LexGrid.LexBIG.cagrid.LexBIGCaGridServices.service.FilterClient object to the client. This is the client to the Filter Service Context. This object has a direct reference to the Resource created above. This FilterClient implements org.LexGrid.LexBIG.Extensions.Query.Filter. The user now uses this client to make transparent Grid calls through the Service Context. Because this Filter implements org.LexGrid.LexBIG.Extensions.Query.Filter, API calls will look to the user as being identical to direct LexBIG API calls.

5. The client may continue to make statefull calls to the FilterClient and the assigned Resource. For example, the client may call any method in org.LexGrid.LexBIG.Extensions.Query.Filter

```
filter.match(resolvedConceptReference);
```

6. These API calls are separate calls but statefully maintained on the server via the Resource.

8. ResolvedConceptReferencesIterator

http://informatics.mayo.edu/LexGrid/downloads/javadoc/org/LexGrid/LexBIG/Utility/Iterators/ResolvedConceptReferencesIterator.html

A ResolvedConceptReferencesIterator is created when a CodedNodeSet or CodedNodeGraph is resolved. It allows results to be returned from the server incrementally instead of all at once. When the user creates a ResolvedConceptReferencesIterator, the server creates and stores the ResolvedConceptReferencesIterator server-side as a Resource. This Resource is associated with the client and will be accessible only by the client that created it.

ResolvedConceptReferencesIterator Call Sequence:

1. The user gets a ResolvedConceptReferencesIterator from a Resolve.

2. The server calls the Distributed LexBIG resolve method on the CodedNodeSet, returning to the server an implementation of org.LexGrid.LexBIG.Utility.Iterators.ResolvedConceptReferencesIterator object.

3. The server then creates an org.LexGrid.LexBIG.cagrid.LexBIGCaGridServices.ResolvedConceptReferencesIterator.service.globus.resource.ResolvedConceptReferencesIteratorResource. This Resource will be used to hold the instance of an implementation of org.LexGrid.LexBIG.Utility.Iterators.ResolvedConceptReferencesIterator.

4. The server returns an org.LexGrid.LexBIG.cagrid.LexBIGCaGridServices.service.ResolvedConceptReferencesIteratorClient object to the client. This is the client to the ResolvedConceptReferencesIterator Service Context. This object has a direct reference to the Resource created above. This ResolvedConceptReferencesIteratorClient implements org.LexGrid.LexBIG.Utility.Iterators.ResolvedConceptReferencesIterator. The user now uses this client to make transparent Grid calls through the Service Context. Because this ResolvedConceptReferencesIterator implements org.LexGrid.LexBIG.Utility.Iterators.ResolvedConceptReferencesIterator, API calls will look to the user as being identical to direct LexEVS API calls.

5. The client may continue to make statefull calls to the ResolvedConceptReferencesIteratorClient and the assigned Resource. For example, the client may call any method in org.LexGrid.LexBIG.Utility.Iterators.ResolvedConceptReferencesIterator

```
while(itr.hasNext){
        ResolvedConceptReference ref = itr.next();
        }
```

6. These API calls are separate calls but statefully maintained on the server via the Resource.

Error Handling

Error Connecting to LexEVS Grid Service

When connecting through the Java Client, java.net.ConnectException and org.apache.axis.types.URI.MalformedURIException may be thrown upon an unsuccessful attempt to connect.

A MalformedURIException is thrown in the case if a poorly-formed URL string. In this case, the exception is thrown before an attempt to connect is even made.

If the URL is well-formed, proper connection is tested. If the connection attempt fails, a ConnectException is thrown containing the reason for the failure.

```
try{
LexBIGServiceGridAdapter lbsg = new LexBIGServiceGridAdapter
("http://localhost:8080/wsrf/services/cagrid/LexEVSGridService");
} catch(java.net.ConnectException e){
        //Error Connecting
        e.printStackTrace();
} catch(org.apache.axis.types.URI.MalformedURIException e){
        //URL Syntax Error
        e.printStackTrace();
}
```

This example shows a typical connection to the LexEVS Grid Service, with the two potential Exceptions being caught and handled as necessary.

LexBIG Errors

LexBIG errors will be forwarded through the Distributed LexEVS layer and then on to the Grid layer. Input parameters, along with any other LexBIG (or Distributed LexBIG) errors will be detected on the server, not the client, and forwarded. All Generic LexEVS (or Distributed LexEVS) errors will be forwarded via a RemoteException, with the cause of the error and underlying LexEVS error message included.

Invalid Service Context Access

Service Context Services are not meant to be called directly. If the client attempts to do so, an org.LexGrid.LexBIG.cagrid.LexEVSGridService.CodedNodeSet.stubs.types.InvalidServiceContextAccess Exception will be thrown. This indicates a call was made to a Service Context without obtaining a Service Context Reference via the Main Service (see the above section Service Contexts and State for more information).

Client

The Introduce toolkit generates a “client” class that will be provided to the users.

Security Issues

Security in the LexEVS Grid Service is implemented in the Distributed LexBIG layer. The information in this section explains how the LexEVS Grid Services utilize this security implementation. For more information about the Distributed LexBIG Security Implementation, see this documentation: http://gforge.nci.nih.gov/tracker/download.php/366/1462/10884/4060/Distributed_LexBIG_%20AccessTo_Licensed_Vocabulary_implemenation.doc

LexEVS Grid Service Security

Certain vocabulary content accessible through the LexEVS Grid Service may require extra authorization to access. Each client is required to supply its own access credentials via Security Tokens. These Security Tokens are implemented by a SecurityToken object: Name: SecurityToken Namespace: gme://caCORE.caCORE/3.2/gov.nih.nci.evs.security Package: gov.nih.nci.evs.security

Accessing Secure Content

A client establishes access to a secured vocabulary via the following Grid Service Calls:

Step 1: Connect to the LexBIG caGrid Service `LexBIGServiceGrid lbs = new LexBIGServiceGridAdapter(url);`

Step 3: Build an org.LexGrid.LexBIG.DataModel.cagrid.CodingSchemeIdentification to hold the Coding Scheme name.

```
CodingSchemeIdentification codingScheme = new CodingSchemeIdentification(); codingScheme.setName(“codingScheme”);
```

Step 4: Build an gov.nih.nci.evs.security.SecurityToken containing the security information for the desired Coding Scheme.

```
SecurityToken token = new SecurityToken(); token.setAccessToken(“securityToken”);
```

Step 5: Invoke the LexBIG caGrid service as follows: This will return a reference to a new “LexBIGServiceGrid” instance that is associated with the security properties that were passed in.

```
LexBIGServiceGrid lbsg = lbs.setSecurityToken(codingScheme, token);
```

It is important to note that the Grid Service “setSecurityToken” returns an *org.LexGrid.LexBIG.cagrid.LexEVSGridService.stubs.types .LexEVSGridServiceReference.LexEVSGridServiceReference* object. This reference must be used to access the secured vocabularies.

Implementation

Each call to “setSecurityToken” sets up a secured connection to Distributed LexBIG with the access privileges included in the SecurityToken parameter. The *LexEVSGridServiceReference* that is returned to the client contains a unique key identifier to the secure connection that has been created on the server. All subsequent calls the client makes through this *LexEVSGridServiceReference* will be made securely. If additional SecurityTokens are passed in through the “setSecurityToken” Grid Service, the additional security will be added and maintained.

The “setSecurityToken” Grid Service is a stateful service. This means that after the client sets a SecurityToken, any subsequent call will be applied to that SecurityToken.

Secure connections are not maintained on the server indefinitely, but are based on load conditions. The server will allow 30 unique secure connections to be set up for clients without any time limitations. As additional requests for secure connections are received by the server, connections will be released by the server on an 'oldest first' basis. No connection, however, may be released prior to 5 minutes after its creation.

If no SecurityTokens are passed in by the client, a non-secure Distributed LexBIG connection will be used. The server maintains one (and only one) un-secured Distributed LexBIG connection that is shared by any client not requesting security.

NOTE:

All non-secured information accessed by the LexEVS Grid Service is publicly available from NCICB and users are expected to follow the licensing requirements currently in place for accessing and using NCI EVS information.

Performance

The LexEVS service will take advantage of all improvements made to the LexEVS API services with the exception of lazy loading. LexEVS grid service, being in nature a web service is currently not taking advantage of lazy loading since objects are transferred as fully populated objects. However, future releases of LexEVS Grid Service may refractor the interface in such as way as to take advantage of some of the benefits brought about by the inclusion of lazy loading in to LexEVS API service.

LexEVS Grid Services utilize the performance enhancements of the LexBIG API. For more information about LexBIG performance (which LexEVS Grid Services are dependent on), see http://informatics.mayo.edu

Installation / Packaging

The service will be installed and deployed as a "stand alone" service at NCICB.

Migration

Both the current version of LexEVS grid service may be "in service" simultaneously if the corresponding underlying EVSAPI service is also "in service" to manage migration of clients.

System Testing

See LexEVS Grid Service Testing Documentation at: http://gforge.nci.nih.gov/docman/index.php?group_id=491&selected_doc_group_id=3879&language_id=1

DOCUMENT APPROVAL

Approvers List

The individuals listed in this section constitute the approvers list for the Integration Test Plan document. Formal approval must be received from all approvers prior to the initiation of the next steps in the process.

TITLE	NAME
Project Manager	
Development Manager	

Reviewers List

The individuals listed in this section constitute the reviewers list for the Master Test Plan document. Formal approval is not required from the reviewers, however, it is desirable to have all reviewers review and comment on the document. Reviewers may choose to concentrate on reviewing only those sections that are in their area of responsibility, rather than the entire document.

TITLE	NAME
Technical Writer	

(DESIGN DOC IMPORT END)

LexEVS Loader Source Mapping

The following documentation is to provide additional detail to how different formats are loaded into the LexEVS model.

Unified Medical Language System

The Unified Medical Language System (UMLS) and Rich Release Format (RRF) files

The UMLS' large medical thesaurus is available as a set of text based, "|' separated files which can be made subset into individual terminologies depending on the user's needs. NCI's MetaThesaurus is also RRF formatted. We map individual terminologies, the entire NCI MetaThesaurus and the UMLS terminology SEMNET into LexGrid Using specific loaders and mappings for each.

Supported Coding Scheme Attributes:

> These aren't mapped as categories to a model element. That is, a supported association has an attributeTag column with a corresponding name, but it's context is implied in the

name of the supported attribute. For instance, supported associations will have an attributeTag of “association” but that tag corresponds to no element in the model element SupportedAssociation. Instead the context is implied in the name of the element SupportedAssociation.

Preferred Presentation Selection:

Preferred Presentation is determined first by sorting the presentations to include first those in the default language of the Terminology. Following that and given there is more than one presentation in the default language the “most preferred” is determined in the following manner:

Using the "isPref" column, the "TS" and "STT" columns in the MRCONSO RRF file, or a combination of these columns. The MRRANK file overrides these columns.

Preferred Definition Selection:

Definitions in UMLs are not ranked, the first definition found for a concept in the source file MRDEF.RRF is set to preferred.

Special SNOMED adjustments for concept presentation language:

Snomed handles it’s language default settings differently than other UMLS terminologies, we hard code it’s default language as “en” as a result.

Presentation language is determined by combining the values of SUI, LUI and CUI from MRCONSO and selecting the ATV value from MRSAT where SAB always equals SNOMEDCT and the ATN value is either LANGUAGECODE or SUBSETLANGUAGECODE.

Association Qualifiers for medDRA and others:

MedDRA employs SMQ’s or Standardized Medical Queries as a method of classifying portions of this terminology. These are expressed in MRSAT.RRF when the AUI in the METAUI column is replaced by a RUI code. In LexBIG is RUI is identified in the MRREL.RRF source as relationships are loaded and the associated ATN and ATV values from the MRSAT.RRF row are populated as association qualifier name and value.

Hierarchies expressed in source contexts:

Hierarchies in the UMLS are expressed in the MRREL.RRF file as source, target pairs. However source hierarchies may also be expressed in the MRHEIR.RRF file. These context based hierarchies are realized in LexBIG by accessing the MRHEIR source where the HCD column value is populate. When this is the case, as in MESH, the path of AUI’s to root from the code in the HCD column is processed as a hierarchy. LexBIG’s behavior is as follows:

- Entries in MRHIER that define multiple contexts (HCD field) per CUI will trigger additional tracking within the LexBIG environment.
- Each link is tracked via the corresponding contextual chain(Path To Root field). To do this, we add association qualifiers that tag the association between each participating concept. The qualifier name is 'HCD' and the value will be the HCD field value from the MRHIER file.
- An individual association between two concepts can participate in multiple context chains by assigning additional association qualifiers. A complete flow across the entire chain of links (essentially reconstructing PTR field) can be derived by recursive evaluation of surrounding links that have the same context qualifications. Since each concept can carry multiple text presentations, property qualifiers will be used to track the individual terms used in each context.
- As with associations, multiple qualifiers can be assigned to each text property. Once again, the qualifier name will be 'HCD' and the value will be the HCD field value from the MRHIER file.
- In order to query context-specific relationships, we can first use the API to filter the relationships a concept participates in, then query neighboring nodes to determine the complete context path, and finally map back to specific terms through the registered HCD qualifiers .

OBO Mapping

The OBO each remark in the document header will be combined and put into the coding scheme entityDescription.

For example:

```
remark: autogenerated-by:     DAG-Edit version 1.320
remark: saved-by:             mariacos
remark: date:                 Fri Jun 27 09:41:28 EDT 2003
remark: version: $Revision: 1.1 $
```

Protege OWL

DatatypeProperty Representation

Owl:

```
<owl:DatatypeProperty rdf:ID="currency">
          <rdfs:domain rdf:resource="#Money"/>
          <rdfs:range rdf:resource="http://www.w3.org/2001/XMLSchema#string"/>
     </owl:DatatypeProperty>
```

In LexGrid, a DatatypeProperty is combination of a conceptProperty and Assocation.

Concept Property

```
<lgCon:concept id="Money">
      <lgCommon:entityDescription>Money</lgCommon:entityDescription>
      ….
     <lgCon:conceptProperty propertyId="P0003" propertyName="currency">
        <lgCommon:text>xsd:string</lgCommon:text>
      </lgCon:conceptProperty>
    </lgCon:concept>
```

Association

```
<lgRel:association id="hasDomain" forwardName="hasDomain"
isReflexive="false" isSymmetric="false"
isTransitive="true" reverseName="kindIsDomainOf">
      <lgRel:sourceConcept sourceEntityType="association"
sourceId="currency">
        <lgRel:targetConcept targetEntityType="concept"
targetId="Money"/>
      </lgRel:sourceConcept>

 <lgRel:association id="currency">
      <associationProperty propertyId="P0007"
propertyName="isDatatypeProperty">
        <lgCommon:text>true</lgCommon:text>
      </associationProperty>
      <associationProperty propertyId="P0008"
propertyName="isObjectProperty">
        <lgCommon:text>false</lgCommon:text>
      </associationProperty>
    </lgRel:association>

 <lgRel:association id="datatype" forwardName="datatype">
      <lgRel:sourceConcept sourceEntityType="association"
sourceId="currency">
        <lgRel:targetDataValue dataId="D0001">
          <lgRel:dataValue>string</lgRel:dataValue>
        </lgRel:targetDataValue>
```

Equivalent Class Representation

Owl:

```
<owl:Class rdf:ID="Father">
    <owl:equivalentClass>
      <owl:Class>
        <owl:intersectionOf rdf:parseType="Collection">
          <owl:Class rdf:about="#Person"/>
          <owl:Restriction>
            <owl:onProperty>
              <owl:FunctionalProperty rdf:about="#hasSex"/>
            </owl:onProperty>
            <owl:hasValue rdf:resource="#MaleSex"/>
          </owl:Restriction>
          <owl:Restriction>
            <owl:someValuesFrom rdf:resource="#Person"/>
            <owl:onProperty>
              <owl:ObjectProperty rdf:about="#hasChild"/>
            </owl:onProperty>
          </owl:Restriction>
        </owl:intersectionOf>
      </owl:Class>
    </owl:equivalentClass>
  </owl:Class>
```

In LexGrid, the equivalentClass is represented as an Association.

Association

```
<lgRel:association id="equivalentClass"
forwardName="equivalentClass" isReflexive="true" isSymmetric="true"
isTransitive="true" reverseName="equivalentClass">
  <lgRel:sourceConcept sourceEntityType="concept" sourceId="Father">
        <lgRel:targetConcept targetEntityType="concept"         targetId="A38"/>
      </lgRel:sourceConcept>
```

Restriction Representation

Owl:

```
<owl:Class rdf:ID="Large-Format">
          <rdfs:subClassOf rdf:resource="#Camera"/>
          <rdfs:subClassOf>
               <owl:Restriction>
                     <owl:onProperty rdf:resource="#body"/>
                     <owl:allValuesFrom rdf:resource="#BodyWithNonAdjustableShutterSpeed"/>
               </owl:Restriction>
          </rdfs:subClassOf>
     </owl:Class>
```

In LexGrid, a restriction is a combination of association and qualifier.

Association:

```
<lgRel:association codingSchemeId="p1" id="body"
forwardName="body" isFunctional="false"
isReverseFunctional="false"
isSymmetric="false" isTransitive="false">
      <lgRel:sourceConcept sourceCodingScheme="p1"
sourceEntityType="concept" sourceId="Large-Format">
        <lgRel:targetConcept targetEntityType="concept"
targetId="BodyWithNonAdjustableShutterSpeed">
          <lgRel:associationQualification
associationQualifier="owl:allValuesFrom"/>
        </lgRel:targetConcept>
      </lgRel:sourceConcept>
      <associationProperty propertyId="P0021"
propertyName="isDatatypeProperty">
        <lgCommon:text>false</lgCommon:text>
      </associationProperty>
      <associationProperty propertyId="P0022"
propertyName="isObjectProperty">
        <lgCommon:text>true</lgCommon:text>
      </associationProperty>
    </lgRel:association>
```

Additional Examples

Owl:

```
<owl:Class rdf:ID="Father">
    <owl:equivalentClass>
      <owl:Class>
        <owl:intersectionOf rdf:parseType="Collection">
          <owl:Class rdf:about="#Person"/>
          <owl:Restriction>
            <owl:onProperty>
              <owl:FunctionalProperty rdf:about="#hasSex"/>
            </owl:onProperty>
            <owl:hasValue rdf:resource="#MaleSex"/>
          </owl:Restriction>
          <owl:Restriction>
            <owl:someValuesFrom rdf:resource="#Person"/>
            <owl:onProperty>
              <owl:ObjectProperty rdf:about="#hasChild"/>
            </owl:onProperty>
          </owl:Restriction>
        </owl:intersectionOf>
      </owl:Class>
    </owl:equivalentClass>
  </owl:Class>
```

LexGrid:

```
<lgRel:association id="equivalentClass"
forwardName="equivalentClass" isReflexive="true"
isSymmetric="true"
isTransitive="true" reverseName="equivalentClass">
  <lgRel:sourceConcept sourceEntityType="concept"
sourceId="Father">
        <lgRel:targetConcept targetEntityType="concept"
targetId="A38"/>
      </lgRel:sourceConcept>


 <lgRel:association codingSchemeId="" id="hasSex"
forwardName="hasSex" isFunctional="true"
isReverseFunctional="false"
isSymmetric="false" isTransitive="false">
    <lgRel:sourceConcept sourceEntityType="concept"
sourceId="A38">
        <lgRel:targetConcept targetEntityType="concept"
targetId="MaleSex">
          <lgRel:associationQualification
associationQualifier="owl:hasValue"/>
        </lgRel:targetConcept>

<lgRel:association codingSchemeId="rdfs" id="subClassOf"
forwardName="subClassOf" isFunctional="false"
isReflexive="true"
isSymmetric="false" isTransitive="true"
reverseName="hasSubClass">
   <lgRel:sourceConcept sourceEntityType="concept"
sourceId="A38">
        <lgRel:targetConcept targetEntityType="concept"
targetId="Person"/>
      </lgRel:sourceConcept>

 <lgRel:association codingSchemeId="" id="hasChild"
forwardName="hasChild" isFunctional="false"
isReverseFunctional="false" isSymmetric="false"
isTransitive="false">
   <lgRel:sourceConcept sourceEntityType="concept"
sourceId="A38">
        <lgRel:targetConcept targetEntityType="concept"
targetId="Person">
          <lgRel:associationQualification
associationQualifier="owl:someValuesFrom"/>
        </lgRel:targetConcept>

<lgCon:concept id="A38" isAnonymous="true">
      <lgCommon:entityDescription>Person and
(hasSex has MaleSex)
and (hasChild some Person)</lgCommon:entityDescription>
      <lgCon:presentation propertyId="P0002"
propertyName="textualPresentation" isPreferred="true">
        <lgCommon:text>Person and (hasSex has MaleSex) and
(hasChild
some Person)</lgCommon:text>
      </lgCon:presentation>
      <lgCon:conceptProperty propertyId="P0001"
propertyName="type">
        <lgCommon:text>owl:intersectionOf</lgCommon:text>
      </lgCon:conceptProperty>
    </lgCon:concept>
```

Property Restriction Representation

Anonymous LexGrid concepts are created for property restrictions (UnionOf, hasValue).

Example 1

Owl:

```
<owl:Class>
        <owl:unionOf rdf:parseType="Collection">
          <owl:Class rdf:about="#Hot"/>
          <owl:Class rdf:ID="Medium"/>
          <owl:Class rdf:about="#Mild"/>
        </owl:unionOf>
      </owl:Class>
```

LexGrid:

```
<lgCon:concept id="A17" isAnonymous="true">
      <lgCommon:entityDescription>Hot or Medium or
Mild</lgCommon:entityDescription>
      <lgCon:presentation propertyId="P0001"
propertyName="textualPresentation" isPreferred="true">
        <lgCommon:text>Hot or Medium or Mild</lgCommon:text>
      </lgCon:presentation>
      <lgCon:conceptProperty propertyId="P0002"
propertyName="isUnion">
        <lgCommon:text>true</lgCommon:text>
      </lgCon:conceptProperty>
      <lgCon:conceptProperty propertyId="P0003"
propertyName="isIntersection">
        <lgCommon:text>false</lgCommon:text>
      </lgCon:conceptProperty>
      <lgCon:conceptProperty propertyId="P0004"
propertyName="isEnumeration">
        <lgCommon:text>false</lgCommon:text>
      </lgCon:conceptProperty>
    </lgCon:concept>
```

Example 2

Owl:

```
           <owl:Restriction>
                <owl:onProperty rdf:resource="#hasTopping"/>
                <owl:allValuesFrom>
                    <owl:Class>
                        <owl:unionOf rdf:parseType="Collection">
                            <owl:Class rdf:about="#MozzarellaTopping"/>
                            <owl:Class rdf:about="#PeperoniSausageTopping"/>
                            <owl:Class rdf:about="#JalapenoPepperTopping"/>
                            <owl:Class rdf:about="#TomatoTopping"/>
                            <owl:Class rdf:about="#HotGreenPepperTopping"/>
                        </owl:unionOf>
                    </owl:Class>
                </owl:allValuesFrom>
            </owl:Restriction>
```

LexGrid:

```
<lgRel:association id="hasTopping" forwardName="hasTopping"
isFunctional="false" isNavigable="true" isReverseFunctional="true"
isSymmetric="false" isTransitive="false">

    <lgRel:sourceEntity sourceCodingScheme="pizza"
sourceEntityType="concept" sourceId="AmericanHot">
        <lgRel:targetEntity targetCodingScheme="pizza"
targetEntityType="concept" targetId="A16">
          <lgRel:associationQualification
associationQualifier="owl:allValuesFrom"/>
        </lgRel:targetEntity>
      </lgRel:sourceEntity>
  </lgRel:association>


        <rdfs:subClassOf>
            <owl:Restriction>
                <owl:onProperty rdf:resource="#hasTopping"/>
                <owl:allValuesFrom>
                    <owl:Class>
                        <owl:unionOf rdf:parseType="Collection">
                            <owl:Class
rdf:about="#MozzarellaTopping"/>
                            <owl:Class
rdf:about="#PeperoniSausageTopping"/>
                            <owl:Class
rdf:about="#JalapenoPepperTopping"/>
                            <owl:Class rdf:about="#TomatoTopping"/>
                            <owl:Class
rdf:about="#HotGreenPepperTopping"/>
                        </owl:unionOf>
                    </owl:Class>
                </owl:allValuesFrom>
            </owl:Restriction>
        </rdfs:subClassOf>


<lgCon:concept id="A16" isActive="true" isAnonymous="true">
      <lgCommon:entityDescription>MozzarellaTopping or
PeperoniSausageTopping or JalapenoPepperTopping or TomatoTopping or
HotGreenPepperTopping</lgCommon:entityDescription>
      <lgCon:presentation propertyId="P0002"
propertyName="textualPresentation" isPreferred="true">
        <lgCommon:text>MozzarellaTopping or PeperoniSausageTopping
or JalapenoPepperTopping or TomatoTopping or
HotGreenPepperTopping</lgCommon:text>
      </lgCon:presentation>
      <lgCon:conceptProperty propertyId="P0001" propertyName="type">
        <lgCommon:text>owl:unionOf</lgCommon:text>
      </lgCon:conceptProperty>
    </lgCon:concept>
```

NCI OWL

Top-level containers for relations are created, which separate the association types based on the notion of ‘associations’ and ‘roles’ as defined by NCI:

- Associations are “non-inheritable, non-defining relations between concepts”
- Roles are “inheritable relationships”

A LexGrid concept is created for every anonymous class present in the OWL ontology.

If no equivalent class for a concept, it is considered primitive and is indicated by creating a concept property set to ‘true.’

Embedded XML

Property text with embedded XML fragments are identified by by the following identifiers:

qual-name qual-value qual

If the extracted tag is one of XML Text identifiers:

Value term-name def-definition go-term

The text of the property is set to the tag value.

If the extracted tag is one of XML Source Name identifiers:

term-source def-source

A property source is created and the tag value identifies the source.

If the property is a presentation and the extracted tag is XML Representational Form:

term-group

The representational form of the presentation property is set to the tag value.

If the extracted tag is one of DB XRef Prefix:

dbxref.*

A property qualifier is created. The property qualifier id is set to the tag, the value is set to the tag value.

HL7 RIM

To build a single coding scheme from the HL7 MS Access database, implementation is similar to how the NCI MetaThesaurus is stored in LexGrid.

For example, here is how entries MTHU021347 and MTHU033458 in ICPC2ICD10ENG (NCI MethThesaurus C1394796) are structured in LexGrid:

Coding Scheme: NCI MetaThesaurus - urn:oid:2.16.840.1.113883.3.26.1.2

Concept Code: C1394796

Entity Description: decompensation; heart, senile

Status: Active

Is Active: true

Is Anonymous: false

Presentation: decompensation; heart, senile

> **Property Name:** textualPresentation
>
> **Property Id:** T-1
>
> **Language:** ENG
>
> **Is Preferred:** true
>
> **Representational Form:** PT
>
> **Source:** ICPC2ICD10ENG , **Role:** null, **SubRef:** null
>
> **Property Qualifier Id:** source-code , **Property Qualifier Content:** MTHU021347

Presentation: heart; decompensation, senile

> **Property Name:** textualPresentation
>
> **Property Id:** T-2
>
> **Language:** ENG
>
> **Is Preferred:** false
>
> **Representational Form:** PT
>
> **Source:** ICPC2ICD10ENG , Role: **null, SubRef: null**
>
> **Property Qualifier Id:** source-code , **Property Qualifier Content:** MTHU033458

ConceptProperty: Mental or Behavioral Dysfunction

> **Property Name:** Semantic_Type
>
> **Property Id:** SemType-1

In HL7, code systems, concepts, and designations are in the following tables:

Table: VCS_concept_code_xref

Internal concept identifier	Code system	OID	Concept code	Case difference Status
10011	2.16.840.1.113883.5.55	M	0	A

10011	2.16.840.1.113883.5.55	R	0	A
10013	2.16.840.1.113883.5.55	RQ	0	A
10014	2.16.840.1.113883.5.55	NP	0	A
10015	2.16.840.1.113883.5.55	NR	0	A
10016	2.16.840.1.113883.5.55	RE	0	A
10017	2.16.840.1.113883.5.55	X	0	A
10019	2.16.840.1.113883.5.57	R	0	A
10020	2.16.840.1.113883.5.57	D	0	A
10021	2.16.840.1.113883.5.57	I	0	A
10022	2.16.840.1.113883.5.57	K	0	A
10023	2.16.840.1.113883.5.57	V	0	A
10025	2.16.840.1.113883.5.57	ESA	0	A
10026	2.16.840.1.113883.5.57	ESD	0	A
10027	2.16.840.1.113883.5.57	ESC	0	A
10028	2.16.840.1.113883.5.57	ESAC	0	A

Table: VCS_concept_designation

Internal Id	Designation	seq - for case differences	language	preferredForLanguage
10011	Mandatory	0	en	-1
10011	Required - V2.x	0	en	0

Query of HL7 internal id, concept code and designation:

codeSystemName	Code system OID	Internal concept identifier	Concept code	Designation
HL7ConformanceInclusion	2.16.840.1.113883.5.55	10011	R	Required - V2.x
HL7ConformanceInclusion	2.16.840.1.113883.5.55	10011	M	Mandatory
HL7ConformanceInclusion	2.16.840.1.113883.5.55	10011	M	Required - V2.x
HL7ConformanceInclusion	2.16.840.1.113883.5.55	10011	R	Mandatory

To represent HL7 in LexGrid:

A single coding scheme will be created in LexGrid.

Each **VCS_concept_code_xref.internalId** will be represented as a LexGrid Concept Code.

The LexGrid Concept Code will be generated by the concatination of **VCS_concept_code_xref.internalId** and **VCS_concept_code_xref.conceptCode2** (separated by a colon ':').

Not only the duplicates that exist within coding schemes will be dealt with using the id/mnemonic concatenation but also those duplicates that exist between coding schemes.

A LexGrid Concept Code Presentation Property will be created for each HL7 designation (VCS_concept_designation).

The Presentation Property will include Presentation (HL7 Designation), Source (HL7 codeSystemName) and a Property Qualifier of source-code (HL7 Concept Code).

For example, the following structure represents both HL7 10011 entries in code system 2.16.840.1.113883.5.55:

Coding Scheme: HL7 - urn:oid:2.16.840.1.113883.3.26.1.2

Concept Code: 10011:M

Entity Description: >The message element must appear every time the message is communicated and its value must not be null. This condition is subject to the rules of multiplicity and conditionality. If a non-null default value is defined for the element, a null value may be communicated.

Status: Active

Is Active: true

Is Anonymous: false

Presentation: Mandatory

> **Property Name:** textualPresentation
>
> **Property Id:** T-1
>
> **Language:** ENG
>
> **Is Preferred:** true
>
> **Representational Form:** PT
>
> **Source:** HL7ConformanceInclusion , **Role:** null, **SubRef:** null
>
> **Property Qualifier Id:** source-code , **Property Qualifier Content:** M

Presentation: Required - V2.x

> **Property Name:** textualPresentation
>
> **Property Id:** T-2

Language: ENG

Is Preferred: false

Representational Form: PT

Source: HL7ConformanceInclusion, **Role:** null, **SubRef:** null

Property Qualifier Id: source-code , **Property Qualifier Content:** M

Coding Scheme: HL7 - urn:oid:2.16.840.1.113883.3.26.1.2

Concept Code: 10011:R

Entity Description: >The message element must appear every time the message is communicated and its value must not be null. This condition is subject to the rules of multiplicity and conditionality. If a non-null default value is defined for the element, a null value may be communicated.

Status: Active

Is Active: true

Is Anonymous: false

Presentation: Mandatory

Property Name: textualPresentation

Property Id: T-1

Language: ENG

Is Preferred: true

Representational Form: PT

Source: HL7ConformanceInclusion , **Role:** null, **SubRef:** null

Property Qualifier Id: source-code , **Property Qualifier Content:** R

Presentation: Required - V2.x

Property Name: textualPresentation

Property Id: T-2

Language: ENG

Is Preferred: false

Representational Form: PT

Source: HL7ConformanceInclusion, **Role:** null, **SubRef:** null

Property Qualifier Id: source-code , **Property Qualifier Content:** R

Loading the HL7 Rim as a monolithic coding scheme

1. Load coding scheme data as HL7 Rim Metadata from the Model table (rather than the coding scheme data for each HL7 coding scheme).
   a. Mapping of these values will be incomplete:
      i. Mapping proposal:

LexGrid	HL7 RIM
<codingSchemeName>	<modelID>
<formalName>	<name>
<registeredName>	http://www.hl7.org/Library/data-model/RIM *
<defaultLanguage>	en*
<representsVersion>	<versionNumber>
<isNative>	0*
<approximateNumberofConcepts>	Result of count on concept bearing table?
<firstRelease>	MISSING
<modifiedInRelease>	MISSING
<deprecated>	MISSING
<entityDescription>	<description>
<copyright>	MISSING

   b. No URN exists and we may need to consider creating one (see entry for registeredName).

2. Locate and load all mappings (such as supportedAssociations and supportedProperties).
   a. Create a supportedHiearchy with a root node of @ on hasSubtype?

3. Iterate through the code system table rows and get each coding scheme.
   a. Create and persist an “@” node in the database
   b. Prepare an artificial “top node“ for each coding scheme. (Metadata persisted here as concept properties?) This will result in 250 top nodes.
      i. The artificial top nodes will need to have a concept code created for them.

ii. Attach to “@” the artificial top nodes as a hasSubtype.
iii. Locate the actual top nodes of each coding scheme by querying the relations table to see if they exist as a target code, if not, they are top nodes so attach them to the artificial top node via hasSubtype.

c. Translate the RRF source property loads to the EMF world.

i. Load the concepts ensuring that the coding scheme name is loaded as a “source” property
ii. Load the relations ensuring that the source and target coding scheme data is loaded with the coding scheme name.

4. Concurrent to this process create an updated “HL7 RIM to LexGrid for NCI” mapping from the current Excel mapping document.

LexGrid Text

The text files that can be imported must use the following formats. Items surrounded by <> are required. Items further surrounded by [] are optional. \t represents a tab - the default delimiter - however other delimiters may be used.

Lines beginning with # are comments.

Format A:

```
<codingSchemeName>\t<codingSchemeId>\t<defaultLanguage>\t<formal
Name>[\t<version>][\t<source>][\t<description>][\t<copyright>]
<name1>[\t <description>]
\t <name2>[\t <description>]
\t\t <name3>[\t <description>]
\t\t <name4>[\t <description>]
```

The leading tabs represent hierarchical “hasSubtype” relationship nesting :

(name1 hasSubtype name2 and name2 hasSubtype name3)

Concept Codes will be automatically generated.

If a name is used more than once - it will be assigned the same code.

If a description is used more than once (for a given name) only the first description will be stored.

Format B:

In this format, concept codes can be provided. This is the same as “Format A” with the inclusion of concept codes as part of the input.

```
<code>\t<name>[\t<description>]
```

If the same code occurs twice, the names must match. Description rules same as “Format A.”

Example of Format A

#lines starting with "#" are comments

#blank lines are ok

#the first "real" line of the file must be of the following format: #<codingSchemeName>\t<codingSchemeId>\t<defaultLanguage>\t<formalName>[\t<version>][\t<source>][\t<description>][\t<copyright>]

colors 1.2.3 en colors coding scheme 1.0 Someone’s Head a simple example coding scheme using colors This isn't worth copyrighting

#The rest of the file (for format A) should look like this:

```
Color   Holder of colors
        Red
        Green   The color Green
                Light Green     foobar
                Dark Green      The color dark green
        Blue
                Red
                        Green   The color Green
```

Example of Format B

#lines starting with "#" are comments

#blank lines are ok

#the first "real" line of the file must be of the following format: #<codingSchemeName>\t<codingSchemeId>\t<defaultLanguage>\t<formalName>[\t<version>][\t<source>][\t<description>][\t<copyright>]

colors2 1.2.4 en colors coding scheme 1.1 Someone’s Head a simple example coding scheme using colors This isn't worth copyrighting

#The rest of the file (for format B) should look like this:

```
1       Color   Holder of colors
        4       Red
        6       Green   The color Green
                7       Light Green
                8       Dark Green
        5       Blue
                8       Dark Green      The color dark green
        6       Green   A different color of green
```

LexEVS Loader Mappings

The following sections give detailed mappings from source formats to LexEVS.

OWL Mapping - 4.2.1

OWL Mapping - Protégé (4.2.1)		
OWL Element	**LexGrid**	**Comments**
OWL: RDF Schema Features		
owl:ontology	codingScheme	
xml:lang	codingScheme.defaultLanguage	Default is 'en'
dc:title	codingScheme.formalName	
rdfs:label	codingScheme.localName	
URI	codingScheme.registeredName	
owl:versionInfo	codingScheme.representsVersion	Default is 'UNASSIGNED'
dc:rights	codingScheme.copyright	
owl:Class (Thing, Nothing)	concept	
rdf:ID	concept.conceptCode	
	concept.isActive	Hard coded as "Active"
	concept.isAnonymous	
rsfs:label	concept.entityDescription	
rdf:comment	concept.comment	
rdfs:subClassOf	association	
	association.id = "subClassOf"	
	association.forwardName = "subClassOf"	
	association.isFunctional = "false"	
	association.isNavigable = "true"	
	association.isReflexive="true"	
	association.isSymmetric="false"	
	association.isTransitive="true"	
rdf:Property (ObjectProperty)	association	An association between two classes (hasDomain, hasRange)
	association concept.conceptProperty	An association between one class (domain) and one asscoication (hasDomain and hasDataProperty). The conceptProperty defines the range.
rdfs:subPropertyOf	association	
	association.id = "subPropertyOf"	
	association.forwardName = "subPropertyOf"	
	association.isFunctional = "false"	
	association.isNavigable = "true"	
	association.isReflexive="true"	
	association.isSymmetric="false"	
	association.isTransitive="true"	
rdfs:domain	association	
	association.id = "hasDomain"	
	association.forwardName = "hasDomain"	
	association.isNavigable = "true"	
	association.isReflexive="false"	
	association.isSymmetric="false"	
	association.isTransitive="true"	
rdfs:range	association	
	association.id = "hasRange"	
	association.forwardName = "hasRange"	
	association.isNavigable = "true"	
	association.isReflexive="false"	
	association.isSymmetric="false"	
	association.isTransitive="false"	
Individual	association	A 'hasInstance' association is created. (ie. sourceId = Country, targetId = America)
	association.id = "hasInstance"	
OWL (In)Equality		
owl:equivalentClass	association	
	association.id = "equivalentClass"	
	association.forwardName = "equivalentClass"	
	association.isFunctional = "false"	
	association.isNavigable = "true"	
	association.isReflexive="true"	
	association.isSymmetric="true"	

	association.isTransitive="true"	
	association.reverseName="equivalentClass"	
owl:equivalentProperty	association	
	association.id = "equivalentProperty"	
	association.forwardName = "equivalentProperty"	
	association.isFunctional = "false"	
	association.isNavigable = "true"	
	association.isReflexive="true"	
	association.isSymmetric="true"	
	association.isTransitive="true"	
	association.reverseName="equivalentProperty"	
owl:sameAs	association	
	association.id = "sameAs"	
	association.forwardName = "sameAs"	
	association.isFunctional = "false"	
	association.isNavigable = "true"	
	association.isReflexive="true"	
	association.isSymmetric="true"	
	association.isTransitive="true"	
	association.reverseName="sameAs"	
differentFrom	association	
	association.id = "differentFrom"	
	association.forwardName = "differentFrom"	
	association.isFunctional = "false"	
	association.isNavigable = "true"	
	association.isReflexive="true"	
	association.isSymmetric="true"	
	association.isTransitive="true"	
	association.reverseName= "differentFrom"	
owl:AllDifferent	association	
	association.id = "AllDifferent"	
	association.forwardName = "AllDifferent"	
	association.isFunctional = "false"	
	association.isNavigable = "true"	
	association.isReflexive="true"	
	association.isSymmetric="true"	
	association.isTransitive="true"	
	association.reverseName= "AllDifferent"	
OWL: Property Characteristics		
owl:inverseOf	association	
	association.id = "inverseOf"	
	association.forwardName = "inverseOf"	
	association.isFunctional = "false"	
	association.isNavigable = "true"	
	association.isReflexive="true"	
	association.isSymmetric="true"	
	association.isTransitive="true"	
	association.reverseName="inverseOf"	
owl:TransitiveProperty	association.isTransitive	association property 'isTransitive'
owl:SymmetricProperty	association.isSymmetric	association property 'isSymmetric'
owl:InverseFunctionalProperty	association.isReverseFunctional	association property 'isReverseFunctional'
owl:FunctionalProperty	association.isFunctional	association property 'isFunctional'
OWL: Property Restrictions		
owl:Restriction	concept	Create an anonymous concept for the restriction
	concept.id	System generated
	concept.isActive = true	
	concept.isAnonymous = true	Hardcoded "True"
owl:onProperty	association.id	
owl: allValuesFrom	concept.entityDescription	String of allValuesFrom values
	concept.presentation.propertyId	Generated value for property textual presentation using "P" concatenated with a steadily incremented numerical value.
	concept.presentation.propertyName	Hardcoded "textualPresentation"

	concept.presentation.isPreferred = true	Hardcoded "true"
	concept.presentation.text	String of allValuesFrom values
	concept.conceptProperty.propertyId	Generated value for property using "P" concatenated with a steadily incremented numerical value.
	concept.conceptProperty.propertyName = type	Hardcoded "type"
	concept.conceptProperty.text = "owl:unionOf"	
owl: someValuesFrom	concept.entityDescription	String of someValuesFrom values
	concept.presentation.propertyId	Generated value for property textual presentation using "P" concatenated with a steadily incremented numerical value.
	concept.presentation.propertyName	Hardcoded "textualPresentation"
	concept.presentation.isPreferred = true	Hardcoded "true"
	concept.presentation.text	String of someValuesFrom values
	concept.conceptProperty.propertyId	Generated value for property using "P" concatenated with a steadily incremented numerical value.
	concept.conceptProperty.propertyName = type	Hardcoded "type"
	concept.conceptProperty.text = "owl:intersectionOf"	
owl:intersectionOf	concept.entityDescription	String of intersectionOf values (ie. Pizza and not VegetarianPizza)
	concept.presentation.propertyId	Generated value for property textual presentation using "P" concatenated with a steadily incremented numerical value.
	concept.presentation.propertyName	Hardcoded "textualPresentation"
	concept.presentation.isPreferred = true	Hardcoded "true"
	concept.presentation.text	String of intersectionOf values (ie. Pizza and not VegetarianPizza)
	concept.conceptProperty.propertyId	Generated value for property using "P" concatenated with a steadily incremented numerical value.
	concept.conceptProperty.propertyName = type	Hardcoded "type"
	concept.conceptProperty.text = "owl:intersectionOf"	
UnionOf	concept.conceptProperty.text = "owl:unionOf"	
owl:complementOf	association	association.id = "subClassOf"
owl:oneOf	concept.entityDescription	String of oneOf values
	concept.presentation.propertyId	Generated value for property textual presentation using "P" concatenated with a steadily incremented numerical value.
	concept.presentation.propertyName	Hardcoded "textualPresentation"
	concept.presentation.isPreferred = true	Hardcoded "true"
	concept.presentation.text	String of oneOf values
	concept.conceptProperty.propertyId	Generated value for property using "P" concatenated with a steadily incremented numerical value.
	concept.conceptProperty.propertyName = type	Hardcoded "type"
	concept.conceptProperty.text = "owl:intersectionOf"	
owl:hasValue	associationQualification.nameAndValueList.content	
owl:minCardinality	concept.entityDescription	String of minCardinality Values (ie. (hasTopping min 3) and Pizza)
	concept.presentation.propertyId	Generated value for property textual presentation using "P" concatenated with a steadily incremented numerical value.
	concept.presentation.propertyName	Hardcoded "textualPresentation"
	concept.presentation.isPreferred = true	Hardcoded "true"
	concept.presentation.text	String of minCardinality Value (ie. (hasTopping min 3) and Pizza)
	concept.conceptProperty.propertyId	Generated value for property using "P" concatenated with a steadily incremented numerical value.
	concept.conceptProperty.propertyName = type	Hardcoded "type"
	concept.conceptProperty.text = "owl:intersectionOf"	
owl:maxCardinality	concept.entityDescription	String of maxCardinality Values (ie. (hasTopping max 2) and Pizza)
	concept.presentation.propertyId	Generated value for property textual presentation using "P" concatenated with a steadily incremented numerical value.
	concept.presentation.propertyName	Hardcoded "textualPresentation"
	concept.presentation.isPreferred = true	Hardcoded "true"
	concept.presentation.text	String of maxCardinality Values (ie. (hasTopping max 2) and Pizza)
	concept.conceptProperty.propertyId	Generated value for property using "P" concatenated with a steadily incremented numerical value.
	concept.conceptProperty.propertyName = type	Hardcoded "type"
	concept.conceptProperty.text = "owl:intersectionOf"	
		String of cardinality Values
owl:cardinality	concept.entityDescription	
	concept.presentation.propertyId	Generated value for property textual presentation using "P" concatenated with a steadily incremented numerical value.
	concept.presentation.propertyName	Hardcoded "textualPresentation"
	concept.presentation.isPreferred = true	Hardcoded "true"
	concept.presentation.text	String of cardinality Values

	concept.conceptProperty.propertyId	Generated value for property using "P" concatenated with a steadily incremented numerical value.
	concept.conceptProperty.propertyName = type	Hardcoded "type"
	concept.conceptProperty.text = "owl:intersectionOf"	
owl:disjointWith	association	association.id = "disjointWith"
OWL: Annotation Property		
rdfs:label	Presentation	
	concept.presentation.propertyId	Generated value for property textual presentation using "P" concatenated with a steadily incremented numerical value.
	concept.presentation.propertyName = "textualPresentation"	Hardcoded "textualPresentation"
	concept.presentation.isPreferred = true	Hardcoded "true"
	concept.presentation.text	Value of rdfs:label
rdfs:comment	Comment	
	concept.comment.propertyId	Generated value for property textual presentation using "P" concatenated with a steadily incremented numerical value.
	concept.comment.propertyName = "comment"	Hardcoded "comment"
	concept.presentation.text	Value of rdfs:comment
rdfs:seeAlso	conceptProperty	
rdfs:isDefinedBy	conceptProperty	
OWL: Versioning		
owl:versionInfo	codingScheme.representsVersion	
priorVersion		Not Mapped
backwardCompatibleWith		Not Mapped
owl:incompatibleWith	association	
	association.id = "incompatibleWith"	
	association.forwardName = "incompatibleWith"	
	association.isFunctional = "false"	
	association.isNavigable = "true"	
	association.isReflexive="true"	
	association.isSymmetric="true"	
	association.isTransitive="true"	
	association.reverseName="incompatibleWith"	
DeprecatedClass	Concept attribute setIsActive = false	Not Mapped
DeprecatedProperty		Not Mapped

OWL Mapping - 5.0

OWL Mapping - Protégé (5.0)		
OWL Element	**LexEVS**	**Comments**
OWL: RDF Schema Features		
owl:ontology	codingScheme	
xml:lang	codingScheme.defaultLanguage	Default is 'en'
dc:title	codingScheme.formalName	
rdfs:label	codingScheme.localName	
URI	codingScheme.registeredName	
owl:versionInfo	codingScheme.representsVersion	Default is 'UNASSIGNED'
dc:rights	codingScheme.copyright	
owl:Class (Thing, Nothing)	concept	
rdf:ID	concept.conceptCode	
	concept.isActive	Hard coded as "Active"
	concept.isAnonymous	
	concept.isDefined	
rsfs:label	concept.entityDescription	
rdf:comment	concept.comment	
rdfs:subClassOf	association	
	association.id = "subClassOf"	
	association.forwardName = "subClassOf"	
	association.isFunctional = "false"	
	association.isNavigable = "true"	
	association.isReflexive="true"	
	association.isSymmetric="false"	

	association.isTransitive="true"	
rdf:Property (ObjectProperty)	association	An association between two classes (domain, range)
	association concept.conceptProperty	An association between one class (domain) and one asscoication (domain and hasDataProperty). The conceptProperty defines the range.
rdfs:subPropertyOf	association	
	association.id = "subPropertyOf"	
	association.forwardName = "subPropertyOf"	
	association.isFunctional = "false"	
	association.isNavigable = "true"	
	association.isReflexive="true"	
	association.isSymmetric="false"	
	association.isTransitive="true"	
rdfs:domain	association	
	association.id = "domain"	
	association.forwardName = "domain"	
	association.isNavigable = "true"	
	association.isReflexive="false"	
	association.isSymmetric="false"	
	association.isTransitive="true"	
rdfs:range	association	
	association.id = "range"	
	association.forwardName = "range"	
	association.isNavigable = "true"	
	association.isReflexive="false"	
	association.isSymmetric="false"	
	association.isTransitive="false"	
Individual	association	An 'instance' association is created. (ie. sourceId = Country, targetId = America)
	association.id = "instance"	
OWL (In)Equality		
owl:equivalentClass	association	
	association.id = "equivalentClass"	
	association.forwardName = "equivalentClass"	
	association.isFunctional = "false"	
	association.isNavigable = "true"	
	association.isReflexive="true"	
	association.isSymmetric="true"	
	association.isTransitive="true"	
	association.reverseName="equivalentClass"	
owl:equivalentProperty	association	
	association.id = "equivalentProperty"	
	association.forwardName = "equivalentProperty"	
	association.isFunctional = "false"	
	association.isNavigable = "true"	
	association.isReflexive="true"	
	association.isSymmetric="true"	
	association.isTransitive="true"	
	association.reverseName="equivalentProperty"	
owl:sameAs	association	
	association.id = "sameAs"	
	association.forwardName = "sameAs"	
	association.isFunctional = "false"	
	association.isNavigable = "true"	
	association.isReflexive="true"	
	association.isSymmetric="true"	
	association.isTransitive="true"	
	association.reverseName="sameAs"	
differentFrom	association	
	association.id = "differentFrom"	
	association.forwardName = "differentFrom"	
	association.isFunctional = "false"	
	association.isNavigable = "true"	
	association.isReflexive="true"	

	association.isSymmetric="true"	
	association.isTransitive="true"	
	association.reverseName= "differentFrom"	
owl:AllDifferent	association	
	association.id = "AllDifferent"	
	association.forwardName = "AllDifferent"	
	association.isFunctional = "false"	
	association.isNavigable = "true"	
	association.isReflexive="true"	
	association.isSymmetric="true"	
	association.isTransitive="true"	
	association.reverseName= "AllDifferent"	
OWL: Property Characteristics		
owl:inverseOf	association	
	association.id = "inverseOf"	
	association.forwardName = "inverseOf"	
	association.isFunctional = "false"	
	association.isNavigable = "true"	
	association.isReflexive="true"	
	association.isSymmetric="true"	
	association.isTransitive="true"	
	association.reverseName="inverseOf"	
owl:TransitiveProperty	association.isTransitive	association property 'isTransitive'
owl:SymmetricProperty	association.isSymmetric	association property 'isSymmetric'
owl:InverseFunctionalProperty	association.isReverseFunctional	association property 'isReverseFunctional'
owl:FunctionalProperty	association.isFunctional	association property 'isFunctional'
OWL: Property Restrictions		
owl:Restriction	concept	Create an anonymous concept for the restriction
	concept.id	System generated
	concept.isActive = true	
	concept.isAnonymous = true	Hardcoded "True"
owl:onProperty	association.id	
owl: allValuesFrom	concept.entityDescription	String of allValuesFrom values
	concept.presentation.propertyId	Generated value for property textual presentation using "P" concatenated with a steadily incremented numerical value.
	concept.presentation.propertyName	Hardcoded "textualPresentation"
	concept.presentation.isPreferred = true	Hardcoded "true"
	concept.presentation.text	String of allValuesFrom values
	concept.conceptProperty.propertyId	Generated value for property using "P" concatenated with a steadily incremented numerical value.
	concept.conceptProperty.propertyName = type	Hardcoded "type"
	concept.conceptProperty.text = "owl:unionOf"	
owl: someValuesFrom	concept.entityDescription	String of someValuesFrom values
	concept.presentation.propertyId	Generated value for property textual presentation using "P" concatenated with a steadily incremented numerical value.
	concept.presentation.propertyName	Hardcoded "textualPresentation"
	concept.presentation.isPreferred = true	Hardcoded "true"
	concept.presentation.text	String of someValuesFrom values
	concept.conceptProperty.propertyId	Generated value for property using "P" concatenated with a steadily incremented numerical value.
	concept.conceptProperty.propertyName = type	Hardcoded "type"
	concept.conceptProperty.text = "owl:intersectionOf"	
owl:intersectionOf	concept.entityDescription	String of intersectionOf values (ie. Pizza and not VegetarianPizza)
	concept.presentation.propertyId	Generated value for property textual presentation using "P" concatenated with a steadily incremented numerical value.
	concept.presentation.propertyName	Hardcoded "textualPresentation"
	concept.presentation.isPreferred = true	Hardcoded "true"
	concept.presentation.text	String of intersectionOf values (ie. Pizza and not VegetarianPizza)
	concept.conceptProperty.propertyId	Generated value for property using "P" concatenated with a steadily incremented numerical value.
	concept.conceptProperty.propertyName = type	Hardcoded "type"
	concept.conceptProperty.text = "owl:intersectionOf"	
UnionOf	concept.conceptProperty.text = "owl:unionOf"	

owl:complementOf	association	association.id = "subClassOf"
owl:oneOf	concept.entityDescription	String of oneOf values
	concept.presentation.propertyId	Generated value for property textual presentation using "P" concatenated with a steadily incremented numerical value.
	concept.presentation.propertyName	Hardcoded "textualPresentation"
	concept.presentation.isPreferred = true	Hardcoded "true"
	concept.presentation.text	String of oneOf values
	concept.conceptProperty.propertyId	Generated value for property using "P" concatenated with a steadily incremented numerical value.
	concept.conceptProperty.propertyName = type	Hardcoded "type"
	concept.conceptProperty.text = "owl:intersectionOf"	
owl:hasValue	associationQualification.nameAndValueList.content	
owl:minCardinality	concept.entityDescription	String of minCardinality Values (ie. (hasTopping min 3) and Pizza)
	concept.presentation.propertyId	Generated value for property textual presentation using "P" concatenated with a steadily incremented numerical value.
	concept.presentation.propertyName	Hardcoded "textualPresentation"
	concept.presentation.isPreferred = true	Hardcoded "true"
	concept.presentation.text	String of minCardinality Value (ie. (hasTopping min 3) and Pizza)
	concept.conceptProperty.propertyId	Generated value for property using "P" concatenated with a steadily incremented numerical value.
	concept.conceptProperty.propertyName = type	Hardcoded "type"
	concept.conceptProperty.text = "owl:intersectionOf"	
owl:maxCardinality	concept.entityDescription	String of maxCardinality Values (ie. (hasTopping max 2) and Pizza)
	concept.presentation.propertyId	Generated value for property textual presentation using "P" concatenated with a steadily incremented numerical value.
	concept.presentation.propertyName	Hardcoded "textualPresentation"
	concept.presentation.isPreferred = true	Hardcoded "true"
	concept.presentation.text	String of maxCardinality Values (ie. (hasTopping max 2) and Pizza)
	concept.conceptProperty.propertyId	Generated value for property using "P" concatenated with a steadily incremented numerical value.
	concept.conceptProperty.propertyName = type	Hardcoded "type"
	concept.conceptProperty.text = "owl:intersectionOf"	
		String of cardinality Values
owl:cardinality	concept.entityDescription	
	concept.presentation.propertyId	Generated value for property textual presentation using "P" concatenated with a steadily incremented numerical value.
	concept.presentation.propertyName	Hardcoded "textualPresentation"
	concept.presentation.isPreferred = true	Hardcoded "true"
	concept.presentation.text	String of cardinality Values
	concept.conceptProperty.propertyId	Generated value for property using "P" concatenated with a steadily incremented numerical value.
	concept.conceptProperty.propertyName = type	Hardcoded "type"
	concept.conceptProperty.text = "owl:intersectionOf"	
owl:disjointWith	association	association.id = "disjointWith"
OWL: Annotation Property		
rdfs:label	Presentation	
	concept.presentation.propertyId	Generated value for property textual presentation using "P" concatenated with a steadily incremented numerical value.
	concept.presentation.propertyName = "textualPresentation"	Hardcoded "textualPresentation"
	concept.presentation.isPreferred = true	Hardcoded "true"
	concept.presentation.text	Value of rdfs:label
rdfs:comment	Comment	
	concept.comment.propertyId	Generated value for property textual presentation using "P" concatenated with a steadily incremented numerical value.
	concept.comment.propertyName = "comment"	Hardcoded "comment"
	concept.presentation.text	Value of rdfs:comment
rdfs:seeAlso	conceptProperty	
rdfs:isDefinedBy	conceptProperty	
OWL: Versioning		
owl:versionInfo	codingScheme.representsVersion	
priorVersion		Not Mapped
backwardCompatibleWith		Not Mapped
owl:incompatibleWith	association	
	association.id = "incompatibleWith"	
	association.forwardName = "incompatibleWith"	

	association.isFunctional = "false"	
	association.isNavigable = "true"	
	association.isReflexive="true"	
	association.isSymmetric="true"	
	association.isTransitive="true"	
	association.reverseName="incompatibleWith"	
DeprecatedClass	Concept attribute setIsActive = false	Not Mapped
DeprecatedProperty		Not Mapped

OWL Mapping - NCI OWL

OWL Mapping - NCI OWL		
OWL Element	**LexGrid**	**Comments**
OWL: RDF Schema Features		
owl:ontology	codingScheme	Hardcoded "NCI_Thesaurus"
xml:lang	codingScheme.defaultLanguage	Hardcoded "en"
dc:title	codingScheme.formalName	Hardcoded "NCI Thesaurus"
rdfs:label	codingScheme.localName	Hardcoded "NCI_Thesaurus"
		Hardcoded "40010"
		Hardcoded "urn:oid:2.16.840.1.113883.3.26.1.1"
URI	codingScheme.registeredName	Hardcoded "http://ncicb.nci.nih.gov/xml/owl/EVS/Thesaurus.owl#"
owl:versionInfo	codingScheme.representsVersion	
dc:rights	codingScheme.copyright	Read from hardcoded "Terms.txt" file .
rdfs:comment	codingScheme.entityDescription	
	codingScheme.isNative	Hardcoded "true"
owl:Class (Thing, Nothing)	concept	
code	concept.id	
	concept.isActive	Hard coded as "true" unless class "owl:DeprecatedClass", then 'false'
	concept.isAnonymous	
rsfs:label	concept.entityDescription	
rdf:comment	concept.comment	
	conceptProperty	Indicate whether the concept is primative (has no equavalent classes)
	concept.conceptProperty.propertyName	Hard coded as "primitive"
	concept.conceptProperty.text	"true"
	concept.conceptProperty.propertyId	Generated value for property using "P" concatenated with a steadily incremented numerical value.
	presentation	Provide default presentation to match concept entity description if not provided as property
	concept.presentation.propertyId	Generated value for property textual presentation using "P" concatenated with a steadily incremented numerical value.
	concept.presentation.propertyName	Hardcoded "NCI_Preferred_Term"
rdfs:label	concept.presentation.text	concept.entityDescription
	conceptProperty	Property with designated concept name label (per NCI requirements and used in codeToName/nameToCode lookup).
	concept.conceptProperty.propertyName	Hard coded as "CONCEPT_NAME"
rdfs:label	concept.conceptProperty.text	concept.entityDescription
	concept.conceptProperty.propertyId	Generated value for property using "P" concatenated with a steadily incremented numerical value.
	relation	Top-level container for associations (non-inheritable, non-defining relationships between concepts.
	relations.dc	Hard coded as "associations"
	relations.isNative	Hard coded as "true"
	relations.entityDescription	Hard coded as "Non-inheritable non-defining relations."
	relation	Top-level container for roles (inheritable relationships)
	relations.dc	Hard coded as "roles"
	relations.isNative	Hard coded as "true"
	relations.entityDescription	Hard coded as "Inheritable/defining relations."
rdfs:subClassOf	association	Association for subtype hierarchy.
	association.id = "hasSubtype"	
	association.forwardName = "hasSubtype"	
	association.reverseName = "isA"	
	association.isNavigable = "true"	Hard coded as "true"
	association.isReflexive="true"	Hard coded as "true"

	association.isSymmetric="false"	Hard coded as "false"
	association.isTransitive="true"	Hard coded as "true"
hasElement	association	Association used to register component classes as elements of anonymous node representations.
	association.id = "hasElement"	
	association.forwardName = "hasElement"	
	association.isNavigable = "true"	Hard coded as "true"
	association.isSymmetric="false"	Hard coded as "false"
	association.isTransitive="true"	Hard coded as "true"
rdfs:domain	association	Association for role_has_domain relations
	association.id = "Role_Has_Domain"	
	association.forwardName = "roleHasDomain"	
	association.reverseName = "kindIsDomainOf"	
	association.isNavigable = "true"	Hard coded as "true"
	association.isReflexive="false"	Hard coded as "false"
	association.isSymmetric="false"	Hard coded as "false"
	association.isTransitive="true"	Hard coded as "true"
rdfs:range	association	Association for range relations
	association.id = "Role_Has_Range"	
	association.forwardName = "roleHasRange"	
	association.reverseName = "kindIsRangeOf"	
	association.isNavigable = "true"	Hard coded as "true"
	association.isReflexive="false"	Hard coded as "false"
	association.isSymmetric="false"	Hard coded as "false"
	association.isTransitive="false"	Hard coded as "false"
rdf:Property (ObjectProperty)	association	An association between two classes (hasDomain, hasRange)
rdfs:subPropertyOf		Not Mapped
OWL (In)Equality		
owl:equivalentClass	association	Association for equivalent class.
	association.id = "equivalentClass"	
	association.forwardName = "equivalentClass"	
	association.reverseName = "equivalentClass"	
	association.isNavigable = "true"	Hard coded as "true"
	association.isReflexive="true"	Hard coded as "true"
	association.isSymmetric="true"	Hard coded as "true"
	association.isTransitive="true"	Hard coded as "true"
OWL: Property Characteristics		
owl:inverseOf	association	
	association.id = "inverseOf"	
	association.forwardName = "inverseOf"	
	association.isFunctional = "false"	
	association.isNavigable = "true"	
	association.isReflexive="true"	
	association.isSymmetric="true"	
	association.isTransitive="true"	
	association.reverseName="inverseOf"	
owl:TransitiveProperty	association.isTransitive	association property 'isTransitive'
owl:SymmetricProperty	association.isSymmetric	association property 'isSymmetric'
owl:InverseFunctionalProperty	association.isReverseFunctional	association property 'isReverseFunctional'
owl:FunctionalProperty	association.isFunctional	association property 'isFunctional'
OWL: Property Restrictions		
owl:Restriction	concept	Anonymous concept created.
	concept.entityDescription = "RestrictionOn: " + association name	Concatination of "Restriction On: " and assocation name
	concept.isAnonymous = true	
owl: allValuesFrom	associationQualification.association.Qualifier = "AllValuesFrom"	
owl: someValuesFrom	associationQualification.association.Qualifier = "someValuesFrom"	
owl:intersectionOf	concept.entityDescription	Concatination of "Restriction On: " and assocation name
	concept.isAnonymous = true	

	concept.presentation.propertyId	Generated value for property textual presentation using "P" concatenated with a steadily incremented numerical value.
	concept.presentation.propertyName	Hardcoded "textualPresentation"
	concept.presentation.isPreferred = true	Hardcoded "true"
	concept.presentation.text	Set to concept.entityDescription
	concept.conceptProperty.propertyId	Generated value for property using "P" concatenated with a steadily incremented numerical value.
	concept.conceptProperty.propertyName = type	Hardcoded "type"
	concept.conceptProperty.text = "owl:intersectionOf"	
owl:unionOf	concept.entityDescription	Concatination of "Restriction On: " and assocation name
	concept.isAnonymous = true	
	concept.presentation.propertyId	Generated value for property textual presentation using "P" concatenated with a steadily incremented numerical value.
	concept.presentation.propertyName	Hardcoded "textualPresentation"
	concept.presentation.isPreferred = true	Hardcoded "true"
	concept.presentation.text	Set to concept.entityDescription
	concept.conceptProperty.propertyId	Generated value for property using "P" concatenated with a steadily incremented numerical value.
	concept.conceptProperty.propertyName = type	Hardcoded "type"
	concept.conceptProperty.text = "owl:unionOf"	
owl:oneOf	concept.conceptProperty.propertyId	Generated value for property using "P" concatenated with a steadily incremented numerical value.
	concept.conceptProperty.propertyName = "owl:oneOf"	Hardcoded "owl:oneOf"
	concept.conceptProperty.text	String of oneOf values
OWL: Annotation Property		
rdfs:comment	Comment	
	concept.comment.propertyId	Generated value for property textual presentation using "P" concatenated with a steadily incremented numerical value.
	concept.comment.propertyName = "comment"	Hardcoded "comment"
	concept.presentation.text	Value of rdfs:comment
rdfs:seeAlso	conceptProperty	
rdfs:isDefinedBy	conceptProperty	
OWL: Versioning		
owl:versionInfo	codingScheme.representsVersion	
priorVersion		Not Mapped
backwardCompatibleWith		Not Mapped
DeprecatedClass		Not Mapped
DeprecatedProperty		Not Mapped

Legacy Complex Prop Mapping

Legacy Complex Properties Mapping							
tag	presentation	source	represenational form	qualifier	model element	value column name	model element
go-term	x					propertyValue	
go-id				x	propertyQualifierId	val1	PropertyQualifier attribute content?
go-source				x	propertyQualifierId	val1	PropertyQualifier attribute content?
source-date				x	propertyQualifierId	val1	PropertyQualifier attribute content?
term-name	x					propertyValue	
term-group			x			representationalForm	property attribute
term-source		x				attributeValue	source
def-source		x				attributeValue	source
def-definition	x					propertyValue	definition
Definition_Review_Date				x	propertyQualifierId	val1	PropertyQualifier attribute content?
Definition_Reviewer_Name				x	propertyQualifierId	val1	PropertyQualifier attribute content?

UMLS SemNet Mapping

UMLS SemNet Mapping					
RRF File Name	***RRF Column Name***	***RRF Definition***	***NCI Meta only***	***LexGrid Model Element***	***comments***
Coding Scheme					

				codingScheme.representsVersion	
				codingScheme.codingScheme	hard coded in java file as "UMLS_SemNet"
				codingScheme.formalName	hard coded in java file as "UMLS Semantic Network"
				codingScheme.defaultLanguage	hard coded in java file as "en"
				codingScheme.approxNumConcepts	hard coded in java file as
				codingScheme.entityDescription	hard coded in java file as "The UMLS Semantic Network is one of three UMLS Knowledge Sources developed as part of the Unified Medical Language System project. The network provides a consistent categorization of all concepts represented in the UMLS Metathesaurus."
license.txt				codingScheme.copyright	Read from license.txt file or hard coded reference in java file
				codingScheme.registeredName	hard coded in java file as "urn:lsid:nlm.nih.gov:semnet"
				codingScheme.concepts.dc	hard coded in java file as "concepts"
				codingScheme.relations.dc	hard coded in java file as "relations"
				codingScheme.mappings.dc	hard coded in java file as "mappings"
				codingScheme.localNameList	
				codingScheme.localNameList.	hard coded in java file as "UMLS_SemNet"
				codingScheme.localNameList	
				codingScheme.localNameList.	
				codingScheme.source	
				codingScheme.source.content	
				codingScheme.localNameList	
				codingScheme.localNameList.	
				codingScheme.localNameList	
				codingScheme.localNameList.	
				codingScheme.localNameList	
				codingScheme.localNameList.	
				codingScheme.localNameList	
				codingScheme.localNameList.	
				mappings.supportedFormat	
				mappings.supportedFormat.localId	hard coded in java file as "text/plain"
				mappings.supportedFormat.urn	hard coded in java file as "urn:oid:2.16.840.1.113883.6.10:text_plain"
				mappings.supportedAssociation	
SRDEF	RL			mappings.supportedAssociation.localId	
				mappings.supportedContext	
				mappings.supportedSource	
				mappings.supportedSource.localId	hard coded in java file as "NLM"
				mappings.supportedSource.urn	hard coded in java file as "urn:lsid:nlm.nih.gov"
				mappings.supportedHierarchy	
				mappings.supportedHierarchy.localId	hard coded in java file as "is_a"
				mappings.supportedHierarchy.isForwardNavigable	hard coded as "true"
				mappings.supportedHierarchy.rootCode	hard coded as "@"
				mappings.supportedHierarchy.associationList	hard coded in java file as "hasSubtype"
				mappings.supportedAssociationQualifier	
SRFLD	COL			mappings.supportedProperty	
				mappings.supportedProperty.localId	If SRDEF appears in the FIL column then this is treated a potential supported property and is entered in supported properties as such.
				mappings.supportedProperty.urn	hard coded in java file as ""
				mappings.supportedLanguage	
				mappings.supportedLanguage.localId	hard coded in java file as "en"
				mappings.supportedLanguage.urn	hard coded in java file as "urn:oid:2.16.840.1.113883.6.84:en"
				mappings.supportedCodingScheme	
				mappings.supportedCodingScheme.localId	hard coded in java file as "UMLS_SemNet"
				mappings.supportedCodingScheme.urn	hard coded in java file as "urn:lsid:nlm.nih.gov:semnet"
				mappings.supportedRepresentationalForm	
				mappings.supportedConceptStatus	
				mappings.supportedPropertyLink	
				mappings.supportedPropertyQualifier	
				mappings.supportedDataType	
Concepts					
SRDEF	UI			concept.id(inherited from Entity)	
SRDEF	STY/RL			concept.enitityDescription(inheritance path Entity->versionableAndDescribable)	
				concept.conceptProperty	
SRDEF	NH			concept.conceptProperty.text.content	
				concept.conceptProperty.format	hard coded in java file as "text/plain"

				concept.conceptProperty.propertyName	hard coded in java file as "NH"
				concept.conceptProperty.propertyId	Generated value for property using "P" concatenated with a steadily incremented numerical value.
				concept.presentation	
				concept.presentation.propertyName (inherited from Property)	Hard coded in java file as "STY/RL" or "ABR"
				concept.presentation.propertyId	Generated value for property using "P" concatenated with a steadily incremented numerical value.
SRDEF	STY/RL, ABR			concept.presentation.text.content	
				concept.presentation.format	hard coded in java file as "text/plain"
				concept.presentation.isPreferred	hard coded in java file as true.
				concept.definition.propertyName (inherited from Property)	Hard coded in java file as "DEF"
				concept.definition.propertyId	Generated value for property using "P" concatenated with a steadily incremented numerical value.
SRDEF	DEF			concept.definition.text.content	
				concept.definition.format	hard coded in java file as "text/plain"
				concept.definition.isPreferred	hard coded in java file as true.
				concept.comment	
SRDEF	EX			concept.comment.propertyName (inherited from Property)	Hard coded in java file as "EX"
				concept.comment.text.content	
				concept.comment.format	hard coded in java file as "text/plain"
				concept.comment.propertyId	Generated value for property using "P" concatenated with a steadily incremented numerical value.
				concept.instruction	
				concept.instruction.propertyName (inherited from Property)	Hard coded in java file as "UN"
SRDEF	UN			concept.instruction.text.content	
				concept.instruction.format	hard coded in java file as "text/plain"
				concept.instruction.propertyId	Generated value for property using "P" concatenated with a steadily incremented numerical value.
Relations					
SRSTR	RL			association.id (inherited from Entity)	In the case of RL value is "isa" the id is hard coded to hasSubtype. The direction of the association is also reversed
				association.isTransitive	hard coded to true if the value of RL is "isa"
SRSTR	RL			association.forwardName	Reversed when value of RL is "isa"
SRSTR	STY/RL			associationInstance.sourceId	Reversed when value of RL is "isa"
SRSTR	STY/RL			associationTarget.targetId	
SRDEF	RIN			association.reverseName	
SRDEF	DEF			association.entityDescription.content (inheritance path for entityDescription is Entity->versionableAndDescribable)	When SRDEF value RT is "RL"
SRSTRE1	UI/STY(first argument)			associationInstance.sourceId	Reversed when value of RL is "isa"
SRSTRE1	UI/STY(2nd argument)			associationTarget.targetId	Reversed when value of RL is "isa"

UMLS Mapping

UMLS Mapping					
RRF File Name	***RRF Column Name***	***RRF Definition***	***NCI Meta only***	***LexGrid Model Element***	***comments***
Coding Scheme					
MRSAB.RRF	SVER	Release date or version number of a source		codingScheme.representsVersion	
MRSAB.RRF	SSN	Source short name		codingScheme.codingScheme	
MRSAB.RRF	SON	Source Official Name		codingScheme.formalName	
MRSAB.RRF	LAT	Language of Term(s)		codingScheme.defaultLanguage	
MRSAB.RRF	TRF	Term frequency for a source		codingScheme.approxNumConcepts	
MRSAB.RRF	SCIT	Source citation		codingScheme.entityDescription	inherits entityDescription from versionableAndDescribable

MRSAB.RRF	SCC	Content contact info for a source		codingScheme.copyright	
				codingScheme.registeredName	Pulled from iso mapping configuration file using method getISOString(RSAB from MRSAB.RRF)
MRDOC.RRF	EXPL	Detailed explanation	x	codingScheme.representsVersion	Where Dockey = "RELEASE" and value = "umls.release.name"
			x	codingScheme.codingScheme	Hard coded in java file as "NCI MetaThesaurus"
			x	codingScheme.formalName	Hard coded in java file as "NCI MetaThesaurus"
			x	codingScheme.defaultLanguage	Hard coded in java file as "ENG"
MRCONSO.RRF			x	codingScheme.approxNumConcepts	Count of CODE value in MRCONSO.RRF
			x	codingScheme.entityDescription	Hard coded in java file as "NCI MetaThesaurus loaded from RRF files."
			x	codingScheme.copyright	Hard coded in java file as "Some material in the NCI Metathesaurus is from copyrighted sources of the respective copyright claimants. All sources appearing in the NCI Metathesaurus are licensed or authorized for NCI use. Users of the NCI Metathesaurus are responsible for compliance with the terms of these licenses and with any copyright restrictions and are referred to NCI Center of Bioinformatics for license terms and to the copyright notices appearing in the original sources, all of which are obtainable online by reference at http://ncimeta.nci.nih.gov/."
				codingScheme.localNameList	Hard coded as constant in java file as "localName"
MRSAB.RRF	SON	Source Official Name		codingScheme.localNameList.	
				codingScheme.localNameList	Hard coded as constant in java file as "localName"
				codingScheme.localNameList.	Pulled from iso mapping configuration file using method getISOString(RSAB from MRSAB.RRF)
				codingScheme.source	Hard coded as constant in java file as "source"
MRDOC.RRF	EXPL	Detailed explanation		codingScheme.source.content	String concatenation of "UMLS-" and value of EXPL
			x	codingScheme.localNameList	Hard coded as constant in java file as "localName"
			x	codingScheme.localNameList.	Hard coded in java file as "NCI Thesaurus"
			x	codingScheme.localNameList	Hard coded as constant in java file as "localName"
			x	codingScheme.localNameList.	Hard coded in java file as "NCI_Thesaurus"
			x	codingScheme.localNameList	Hard coded as constant in java file as "localName"
			x	codingScheme.localNameList.	Hard coded in java file as "10001"
			x	codingScheme.localNameList	Hard coded as constant in java file as "source"
			x	codingScheme.localNameList.	Hard coded in java file as "RRF Files"
				mappings.supportedFormat	Hard coded as constant in java file as "Format"
				mappings.supportedFormat.localId	Hard coded as one of several constants in a java file
				mappings.supportedAssociation	Hard coded as constant in java file as "Association"
MRREL.RRF	REL, RELA	Relationship, Relationship attribute		mappings.supportedAssociation.localId	
				mappings.supportedContext	Hard coded as constant in java file as "Context" May not be used in individual RRF load
				mappings.supportedSource	Hard coded as constant in java file as "Source" May not be used in individual RRF load
				mappings.supportedHierarchy	Hard coded as constant in java file as "Hierarchy"
				mappings.supportedAssociationQualifier	Hard coded as constant in java file as "AssociationQualifier"
				mappings.supportedProperty	Hard coded as constant in java file as "Property"
				mappings.supportedLanguage	Hard coded as constant in java file as "Language"
				mappings.supportedCodingScheme	Hard coded as constant in java file as "CodingScheme"
				mappings.supportedRepresentationalForm	Hard coded as constant in java file as "RepresentationalForm"
				mappings.supportedConceptStatus	Hard coded as constant in java file as "ConceptStatus"
				mappings.supportedPropertyLink	Hard coded as constant in java file as "PropertyLink"
				mappings.supportedPropertyQualifier	Hard coded as constant in java file as "PropertyQualifier"
				mappings.supportedDataType	Hard coded as constant in java file as "DataType"
Concepts					
MRCONSO.RRF	CODE	Unique Identifier or code for string in source		concept.conceptCode	
MRCONSO.RRF	CUI	Unique identifier for concept	x	concept.conceptCode	

				concept.isActive	Hardcoded in parameter as true.
				concept.conceptStatus	Hard coded as constant in java file as "Active"
				concept.isAnonymous	Hardcoded in parameter as false.
MRCONSO.RRF	STR	String		concept.entityDescription	
				concept.conceptProperty.Format	Hard coded as constant in java file as "text/plain" or null
				concept.conceptProperty.propertyName	May be hard coded as constant in java file as one of several properties.
				concept.conceptProperty.usageContext	
				concept.conceptProperty.propertyId	Generated value for property using "P" concatenated with a steadily incremented numerical value.
				concept.presentation.propertyId	Generated value for property textual presentation using "T" concatenated with a steadily incremented numerical value.
				concept.comment.propertyId	Generated value for property comment using "C" concatenated with a steadily incremented numerical value.
				concept.definition.propertyId	Generated value for property definition using "D" concatenated with a steadily incremented numerical value.
				concept.instruction.propertyId	Generated value for property instruction using "I" concatenated with a steadily incremented numerical value.
MRCONSO.RRF	CUI	Unique identifier for concept		concept.conceptProperty.text.content.	
				concept.conceptProperty.propertyId	Generated value for property using "CUI" concatenated with a steadily incremented numerical value.
				concept.conceptProperty.propertyName	hard coded as constant in java file as "UMLS_CUI"
				concept.conceptProperty.propertyType	hard coded as constant in java file as "property"
				concept.conceptProperty.format	left as null
MRSTY.RRF	STY	Semantic type		concept.conceptProperty.text.content	
				concept.conceptProperty.propertyId	Generated value for property using "SemType" concatenated with a steadily incremented numerical value.
				concept.conceptProperty.propertyName	hard coded as constant in java file as "Semantic_Type"
				concept.conceptProperty.propertyType	hard coded as constant in java file as "property"
				concept.conceptProperty.format	Hard coded as constant in java file as "text/plain"
MRCONSO.RRF	LAT	Language of Term(s)		concept.conceptProperty.language	Logic of code simply selects the first definition in the source as the preferred source
MRCONSO.RRF	TS	Term status		concept.presentation.isPreferred	One or a combination of these RRF values determines whether a presentation is preferred: LAT, TS, STT, ISPREF, RANK.
MRCONSO.RRF	STT	String type		concept.presentation.isPreferred	One or a combination of these RRF values determines whether a presentation is preferred: LAT, TS, STT, ISPREF, RANK.
MRCONSO.RRF	ISPREF	Indicates whether AUI is preferred		concept.presentation.isPreferred	One or a combination of these RRF values determines whether a presentation is preferred: LAT, TS, STT, ISPREF, RANK.
MRRANK.RRF	RANK	Termgroup ranking		concept.presentation.isPreferred	One or a combination of these RRF values determines whether a presentation is preferred: LAT, TS, STT, ISPREF, RANK.
				concept.presentation.isPreferred	The first presentation for each language is automatically marked as isPreferred="true" after using comparator to sort list of presentations using comparator to evaluate each presentation based on a combination of values from LAT, TS, STT, ISPREF, RANK.
MRDEF.RRF	DEF	Definition		concept.definition.text.content	
				concept.definition.isPreferred	Logic of code simply selects the first definition in the source as the preferred source
MRSAT.RRF	ATN	Attribute name		concept.conceptProperty.propertyType	Translated to a LexGrid property type. For values AN, CX, HN this property is typed as a "COMMENT" in LexGrid. For value EV this property is typed "PRESENTATION" This only occurs when the STYPE points to the CODE, SCUI or SDUI columns in MRREL.RRF or MRCONSO.RRF. If the STYPE points to SAUI then the values are loaded as property qualifiers.
MRSAT.RRF	ATV	Attribute value		concept.conceptProperty.propertyValue	

MRSAT.RRF	ATN	Attribute name		concept.conceptProperty.propertyQualifier.propertyQualifierId	If the STYPE points to SAUI then the value is loaded as a property qualifier attribute
MRSAT.RRF	ATV	Attribute value		concept.conceptProperty.propertyQualifier.content	If the STYPE points to SAUI then the value is loaded as a property qualifier attribute
MRCONSO.RRF	SAB		x	concept.conceptProperty.source.content	
			x	concept.conceptProperty.propertyQualifier.propertyQualifierId	hard coded as constant in java file as "source-code"
MRCONSO.RRF	CODE		x	concept.conceptProperty.propertyQualifier.content	
			x	concept.conceptProperty.propertyQualifier.propertyQualifierId	hard coded as constant in java file as "AUI"
MRCONSO.RRF	AUI		x	concept.conceptProperty.propertyQualifier.content	
				concept.presentation.representationalForm	When ATN value is EV this presentation will be given a representationalForm of "Abbrev."
MRCONSO.RRF	TTY	Term type in source		concept.presentation.representationForm	When TTY value is FN then representationalForm is represented as "Full Form" Otherwise the representationalForm is the same as the TTY source (i.e. if TTY is PT then representationalForm is PT.) PT is one of the preferred presentations.
				concept.conceptProperty.propertyQualifier.propertyQualifierId	hard coded as "HCD"
MRHIER.RRF	HCD	Source asserted hierarchical number or code for this atom in this context		concept.conceptProperty.propertyQualifier.content	This propertyQualifier is present when the HCD is populated in the the MRHIER file. The corresponding code and property for concept or code is qualified as a code or concept with a context derived heirarchy.
Relations					
MRREL.RRF	CUI1	Unique identifier for first concept			
MRREL.RRF	AUI1	Unique identifier for first atom			
MRCONSO.RRF	CODE	Unique Identifier or code for string in source		ConceptReference.conceptCode (Model element is a ResolvedConceptReference with the value sourceOf attached to the appropriate AssociationList containing this particular REL or RELA association name.)	Mapping to the CODE depends upon the CUI or a combination of CUI and AUI values. If the CODE value is "NOCODE" then LexBIG concatenates "NOCODE" with a "-" and the CUI value. Target or source code value requires use of the DIR flag which indicates the directionality of the relationship in REL or RELA. CUI1 can be used as a pointer to the source CODE value if DIR equals Y, else CUI1 is the targetCode. Similarly, if an AUI exists AUI1 can be an indicator for CODE value to be either or source or target depending on the DIR flag.
MRREL.RRF	CUI2	Unique identifier for second concept			
MRREL.RRF	AUI2	Unique identifier for second atom			
MRCONSO.RRF	CODE	Unique Identifier or code for string in source		ConceptReference.conceptCode (Model element is a ResolvedConceptReference with the value targetOf attached to the appropriate AssociationList containing this particular REL or RELA association name.)	Mapping to the CODE depends upon the CUI or a combination of CUI and AUI values. If the CODE value is "NOCODE" then LexBIG concatenates "NOCODE" with a "-" and the CUI value. Target or source code value requires use of the DIR flag which indicates the directionality of the relationship in REL or RELA. CUI2 can be used as a pointer to the source CODE value if DIR equals Y, else CUI1 is the targetCode. Similarly, if an AUI exists AUI2 can be an indicator for CODE value to be either or source or target depending on the DIR flag.
MRREL.RRF	DIR	Source asserted directionality flag			The UMLS directional flag. Y indicates that this is the direction of the RELA relationship in its source; N indicates that it is not; otherwise indicates that it is not important or has not yet been determined. (If blank RELA, we interpret as 'N', based on empirical review of meta files).
MRREL.RRF	RELA	Relationship attribute		association.id (id inherited from Entity)	Source defined associations. If RELA value is "inverse_isa" then it is changed to "hasSubtype." All others mapped as defined in source.
MRREL.RRF	REL	Relationship		association.id (id inherited from Entity)	UMLS defined associations
MRSAT.RRF	METAUI	Metathesaurus asserted unique identifier			Presence of RUI in MRSAT.RRF METAUI column indicates the association defined in MRREL has an association qualifier. Currently only MedDRA uses these.
MRSAT.RRF	ATN			AssociatedConcept.nameAndValueList.name	
MRSAT.RRF	ATV			AssociationQualification.nameAndValueList.content	
				AssociatedConcept.nameAndValueList.name	qualifier name is hard coded to "HCD" This association qualifier is attached to an association when the HCD field in MRHIER.RRF is populated. Associations are identified by evaluating a structured

					series of AUI's that describe the path to root (PTR field in MRHIER) Once these associations are identified they have and association qualifier attached to them with the value of the HCD loaded as the qualifier.
MRHIER.RRF	HCD			AssociationQualification.nameAndValueList.content	
MRSAB.RRF	SSN	Source short name		association.codingSchemeId (Inherited from Entity)	
MRREL.RR	REL or RELA	Relationship or Relationship attribute		association.forwardName	unqualified REL or RELA value (inverse_isa remains the same)
MRDOC.RRF	EXPL	Detailed explanation		association.reverseName	Where DOCKEY in MRDOC equals REL or RELA and value is the association name and TYPE is REL or RELA name prepended to "_inverse".
				association.inverse	Hard coded as a blank string.
				association.isAntiReflexive	hard coded to null.
				association.isAntiSymmetric	hard coded to null.
				association.isAntiTransitive	hard coded to null.
				association.isAntiTransitive	hard coded to null.
				association.isNavigable	hard coded as Boolean with value true.
				association.isReflexive	hard coded to null.
				association.isReverseFunctional	hard coded to null.
				association.isSymmetric	hard coded to null.
MRREL.RRF	SAB, REL, RELA	Source abbreviation		association.isTransitive	True when the name of the association can be mapped to a source defined in the SAB attribute of MRREL.RRF. Not the SAB value itself, but extrapolated from it using SAB to REL, RELA relationship.
				association.isTranslationAssociation	hard coded to null.
				association.targetCodingScheme	hard coded to null.
				association.entityDescription.content (inheritance path for entityDescription is Entity->versionableAndDescribable)	Hard coded to: "UMLS-defined relationships"
				relations.dc	If REL, this is hard coded as "UMLS-Relations" if RELA then it is hard coded to "Relations"
MRREL.RRF	REL, RELA		x	propertyLink.link	This is a link established when the MRREL.RRF file contains a relationship where the CUI is related to itself. Under these conditions the relationship is mapped as a property link with the MRREL defined relationship mapped as the link value.
			x	propertyLink.sourceProperty	Generated as a propertyId for concept, ex: "T-10" This is retrieved based on the AUI value in MRCONSO.RRF from the entityPropertyMultiAttrib table where the AUI equals the attributeValue column.
			x	propertyLink.targetProperty	Generated as a propertyId for concept, ex: "T-10" This is retrieved based on the AUI value in MRCONSO.RRF from the entityPropertyMultiAttrib table where the AUI equals the attributeValue column.

SNOMED UMLS Mapping

SNOMED UMLS Mapping				
RRF File Name	**RRF Column Name**	**RRF Definition**	**LexGrid Model Element**	**comments**
RSAB.RRF	SVER	Release date or version number of a source	codingScheme.representsVersion	
RSAB.RRF	SSN	Source short name	codingScheme.codingScheme?	
RSAB.RRF	SON	Source Official Name	codingScheme.formalName	
		Hard coded to "en"	codingScheme.defaultLanguage	
MRSAT.RRF	ATV		concept.presentation.language	Unique to snomed.

OBO Mapping

OBO Mapping			
OBO Class	**OBO Entity**	**LexGrid Model Element**	**Notes**
Document Header	format-version		Not mapped.
Document Header	data-version	CodingScheme.representsVersion	Creates a codingSchemeVersion and SystemRelease record. If not specified, then hard coded "UNASSIGNED"

Document Header	version	CodingScheme.representsVersion	Deprecated - use data-version if present.
Document Header	date		Not mapped.
Document Header	saved-by		Ignored but included if contained in the remark entity.
Document Header	auto-generated-by		Ignored but included if contained in the remark entity.
Document Header	subsetdef		Not mapped.
Document Header	import		Deprecated - Imports are used to assemble a larger document from smaller.
Document Header	typeref		Deprecated.
Document Header	synonymtypedef		Not mapped.
Document Header	idspace		Not mapped.The idspace is a triple - localName, URN and description.
Document Header	default-relationship-id-prefix		Not mapped.
Document Header	id-mapping	CodingScheme.supportedAssociation	This is more generalized than the LexGrid model, as it supports mapping between *any* id's. Note that its primary purpose, however, is to handle supportedAssociation.
Document Header	remark	CodingScheme.entityDescription	Will combine multiple remark entities into the entityDescription.
Document Header	default-namespace	codingScheme.codingScheme	Will use default-namespace if provided; otherwise will use filename without the extension.
Document Header	default-namespace	codingScheme.formalName	Will use default-namespace if provided; otherwise will use filename without the extension.
Document Header	default-namespace	codingScheme.registeredName	Combination of "urn:lsid:bioontology.org:" and if provided, the value in "default-namespace"; but if not will use filename without the extension.
		codingScheme.defaultLanguage	Hardcoded "en"
		codingScheme.isNative	Hardcoded "true"
Stanza	id	CodedEntry.conceptCode	
Stanza	name	CodedEntry.entityDescription	
		CodedEntry.presentation['textualPresentation'].text	
		CodedEntry.presentation['textualPresentation'].isPreferred = true	
Stanza	alt_id	CodedEntry.property.property="alt_id"	
		CodedEntry.property['alt_id'].propertyId	
		CodedEntry.property['alt_id'].text	
Stanza	is_anonymous	CodedEntry.isAnonymous = true	
Stanza	is_obsolete	CodedEntry.isActive = false	
Stanza	def	CodedEntry.definition	
		CodedEntry.definition.isPreferred = true	
Stanza	def.dbxref		See dbxref
Stanza	comment	CodedEntry.comment.text	
Stanza	subset	property[subset tag]	See subsetdef
Stanza	syonym	presentation['textualPresentation'].text	
Stanza	synonym.scope	presentation['textualPresentation'].degreeOfFidelity	
Stanza	synonym.type	presentation['textualPresentation'].representationalForm	
Stanza	synonym.dbxref	(see dbxref)	
Stanza	exact_synonym		See synonym
Stanza	narrow_synonym		See synonym
Stanza	broad_synonym		See synonym
Stanza	xref	associations.['mapsTo']	
Stanza	xref_analog		See synonym
Stanza	xref_unk		
Stanza	is_a	associations.['hasSubtype']	Reverse of the source and target.
Stanza	is_a.namespace		If present, the supplied namespace becomes the owning "codingScheme".
Stanza	is_a.derived	associations.hasSubtype.associationQualifier	If present, need to include derived in the supportedAssociationQualifiers section
Stanza	intersection_of		Processed the same way that OWL intersection operator is processed. This includes creation of anonymous sets.
Stanza	union_of		Same as OWL
Stanza	disjoint_from		Same as OWL
Stanza	relationship	associations.	
Stanza	relationship.not_necessary	associations..associationQualifier	

Stanza	relationship.inverse_necessary	associations..associationQualifier	
Stanza	relationship.namespace		If present, the supplied namespace becomes the owning "codingScheme".
Stanza	relationship.derived	associations..associationQualifier	
Stanza	relationship.cardinality	associations..associationQualifier	
Stanza	relationship.maxCardinality	associations..associationQualifier	
Stanza	relationship.minCardinality	associations..associationQualifier	
Stanza	is_obsolete	codedEntry.isActive = false	
		codedEntry.conceptStatus="is_obsolete"	
Stanza	replaced_by		
Stanza	consider		Not Mapped
Stanza	use_term		(deprecated)
dbxref	dbxref name	CodedEntry..source	
		supportedSource	dbxref name format is inconsistent. In most cases, it can be the localName of supportedSource, but special processing may be necessary in the case of URL's, etc
dbxref	dbxref description		Not mapped.
dbxref	trailing modifiers		Not mapped.
typeDef Stanza	domain	associations.['has_domain']	
typeDef Stanza	range	associations.['has_range']	
typeDef Stanza	is_cyclic	property['is_cyclic']	
typeDef Stanza	is_reflexive	property['is_reflexive']	
		association.isReflexive	
typeDef Stanza	is_symmetric	property['is_symmetric']	
		association.isSymmetric	
typeDef Stanza	is_transitive	property['is_transitive']	
		association.isTransitive	
typeDef Stanza	inverse_of	association.inverse	
instance stanza	id	same rules as general stanza	same rules as general stanza
instance stanza	name	same rules as general stanza	same rules as general stanza
instance stanza	instance_of	association['has_instance']	
instance stanza		CodedEntry.property.property=""	data type properties go in Coded Entry property section

HL7 RIM Mapping

		HL7 RIM Mapping
HL7 Table	**HL7 Column**	**LexGrid Model Element**
Model	<modelID>	<codingSchemeName>
	<name>	<formalName>
		<registeredName>
		<defaultLanguage>
	<versionNumber>	<representsVersion>
		<isNative>
		<approximateNumberofConcepts>
		<firstRelease>
		<modifiedRelease>
		<deprecated>
	<description>	<entityDescription>
		<copyright>
VCS_code_system	codeSystemId	codingScheme.registeredName
	codeSystemType	commonTypes::Properties

	codeSystemName	concept.conceptCode
	codeSystemName	concept.presentation['textualPresentation'].text
	fullName	codingScheme.formalName
	description	codingScheme.entityDescription
	releaseId	codingScheme.representsVersion
	copyrightNotice	codingScheme.copyright
	literal('en')	codingScheme.defaultLanguage
VCS_concept_code_xref		
	Concept Code	concept.conceptCode
	Case Difference	commonTypes::Properties
	Status	concept.isActive=(conceptStatus=='A'?)
		concept.conceptStatus
VCS_concept_designation		
	designation	concept.presentation['textualPresentation'].text
	language	concept.presentation['textualPresentation'].language
	preferredForLanguage	concept.presentation['textualPresentation'].isPreferred
	internalId	with(codeSystem[deref(internalId)].concept[deref(internalId)]).definition
	description	concept.presentation['textualPresentation'].text
	language	concept.presentation['textualPresentation'].language
	literal('true')	concept.presentation['textualPresentation'].isPreferred
	uniqueId()	concept.presentation['textualPresentation'].propertyId
	literal('definition')	concept.presentation['textualPresentation'].property
VCS_concept_property	internalId	
	propertyCode	concept.property.property
	propertySeq	
	propartyValue	concept.property.text
	language	concept.property.language
VCS_concept_relationship	relationCode	association.association
	sourceInternalId	associationInstance.sourceConcept
	targetInternalId	associationTarget.targetConcept
Model	modelID	systemRelease.releaseId
	name	service.service
	versionNumber	service.version
	lastModifiedDate	systemRelease.releaseDate
	developingOrganization	systemRelease.releaseAgency
	committeeID	
	description	systemRelease.entityDescription
	concat('urn:oid:2.16.840.1.113883:',systemRelease.releaseId)	systemRelease.releaseURN
	literal('true')	systemRelease.isLatest
	preceding-sibling/releaseOrder + 1	systemRelease.releaseOrder
Model	modelID	commonTypes::Properties
(Special mapping for NCI)	name	codingScheme.localName
	versionNumber	codingScheme.representsVersion
	lastModifiedDate	commonTypes::Properties
	developingOrganization	commonTypes::Properties
	committeeID	
	description	codingScheme.entityDescription
	concat('urn:oid:2.16.840.1.113883:',systemRelease.releaseId)	codingScheme.registeredName
	literal('true')	commonTypes::Properties
	preceding-sibling/releaseOrder + 1	commonTypes::Properties
RIM_vocabulary_domain	vocDomain	codingscheme["VocabularyDomain"].concept.conceptCode

		codingscheme["VocabularyDomain"].concept.presentation["textualPresentation"].
	description	codingscheme["VocabularyDomain"].concept.definition.text
	restrictsDomain	codingscheme["VocabularyDomain"].association["hasSubtype"].sourceConcept
		codingscheme["VocabularyDomain"].association["hasSubtype"].targetconcept = v
VOC_code_reference	usedToBuildValueSet	with(valueDomain[registeredName=current()/.])
	referencesConceptCode	
	referencesInternalId	
	relationship	…valueDomainEntry/includeChildren = (relationship == 'hasSubtype')
	includeReferencedCode	…valueDomainEntry/isSelectable
	leafOnly	
	directChildrenOnly	
	isHeadCode	
	referencesCodeSystem	…/valueDomainEntry.codingScheme
	arbitraryUniqueValue()	…/valueDomainEntry.id
	codeSystemId	
	sponsor	
	publisher	
	versionReportingMethod	
	licensingInformation	
	inUMLS	
	systemSpecificLocatorInfo	
	uri	
	isExternal	
VOC_value_set	valueSetId	valueDomain.registeredName
	valueSetName	valueDomain.valueDomain
	basedOnCodeSystem	valueDomain.defaultCodingScheme
	description	valueDomain.entityDescription
	definingExpression	
	allCodes	if 'true': valueDomain.conceptCode = "@", valueDomain.includeChildren='true'
	isTaxonomicSet	
	valueSetAuthority	
	valueSetNumber	
VOC_value_set_constructor	usedToBuildValueSet	new valueDomainEntry(parent = valueDomain[valueSetId=current()/.],id=unique(
	includesOrExcludesSet	valueDomainEntry.includesValueDomain
	includeHeadCode	valueDomainEntry.isSelectable
		valueDomainEntry.conceptCode = VOC_code_reference[usedToBuildValueSet=c and isHeadCode=true].referencesConceptCode
VOC_vocabulary_domain_value_set	representsVocDomain	(selector)
	definedByValueSet	codingscheme['VocabularyDomain'].concept[representsVocDomain].property['def
	appliesInContext	codingscheme['VocabularyDomain'].concept[representsVocDomain].property['def
VCS_release_version	releaseId	codingSchemeVersion.version
		valueDomainVersion.version
	literal("false")	codingSchemeVersion.isComplete

	releaseAgency	
	releaseDate	codingSchemeVersion.versionDate
		valueDomainVersion.versionDate
	description	codingSchemeVersion.entityDescription
		valueDomainVersion.entityDescription
	editorID	
	forWhomID	
	concat('urn:oid:2.16.840.1.113883:',systemRelease.releaseId)	

LexGrid Text Mapping

LexGrid Text Mapping											
		Source Definition									Comments
		Column	1	2	3	4	5	6	7	8	
Line	1		<codingSchemeName>	<codingSchemeId>	<defaultLanguage>	<formalName>	[<version>]	[<source>]	[<description>]	[<copyright>]	This must be the first line. It contains the coding scheme metadata.
	2		[<code>]	<name>	[<description>]						Beginning of concepts in coding scheme.
	3			[<code>]	<name>	[<description>]					Represent hierarchical 'hasSubtype' relationship nesting (name hasSubtype name)
			Text Element	**LexGrid**	**Comments**						
			Coding Scheme								
			codingSchemeName	codingScheme.codingSchemeName							
			codingSchemeId	codingScheme.codingSchemeId							
			defaultLanguage	codingScheme.defaultLanguage							
			formalName	codingScheme.formalName							
			version	codingScheme.representsVersion	Optional						
			source	codingScheme.source	Optional						
			description	codingScheme.entityDescription	Optional						
			copyright	codingScheme.copyright	Optional						
			Concepts								
			code	concept.conceptCode	Optional						
			name	concept.conceptName							
			description	concept.entityDescription							

Retrieved from "https://cabig-kc.nci.nih.gov/Vocab/KC/index.php/LexEVS_5.0_Design_and_Architecture_Guide"

- This page was last modified on 18 May 2009, at 17:43.

CONTACT USPRIVACY NOTICEDISCLAIMERACCESSIBILITYAPPLICATION SUPPORT