LexEVS 6.0 Design Document - Detailed Design - Authoring

Contents of this Page

Document Information

Author: Craig Stancl
Email: Stancl.craig@mayo.edu
Team: LexEVS

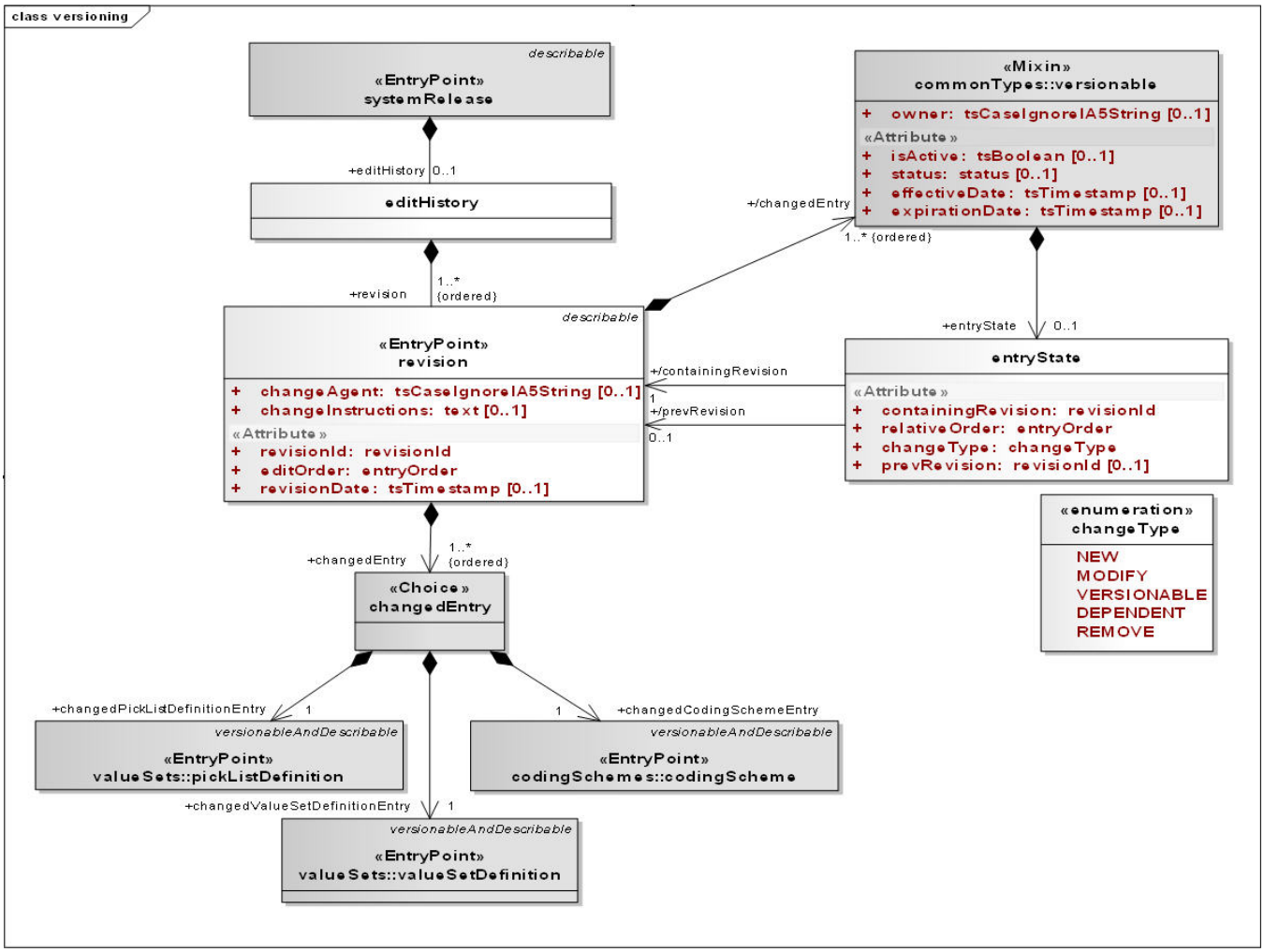
Contract: CBITT BOA Subcontract# 29XS223
Client: NCI CBIIT
National Institutes of Heath
US Department of Health and Human Services

Revision History

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

LexEVS API and the LexGrid model have been designed to provide capabilities to make changes to the terminology elements such as code system or value sets. However, the current LexGrid architecture and model is based on the premise that the information being provided is valid and consistent and is not designed to support partially formed artifacts such as concepts without associated codes, associations that have a source but no target, etc.
The LexEVS authoring tasks assumes that there is an external authoring tool that persists partially formed content and performs the necessary validation and reasoning tasks prior to their being incrementally loaded into the LexEVS services. We see this as being a necessary separation, as the potential combination of editors, reasoners, terminology models, etc. is almost limitless, and each of these will have its own requirements when it comes to completeness and validity.

LexGrid versioning model



LexGrid versioning model description

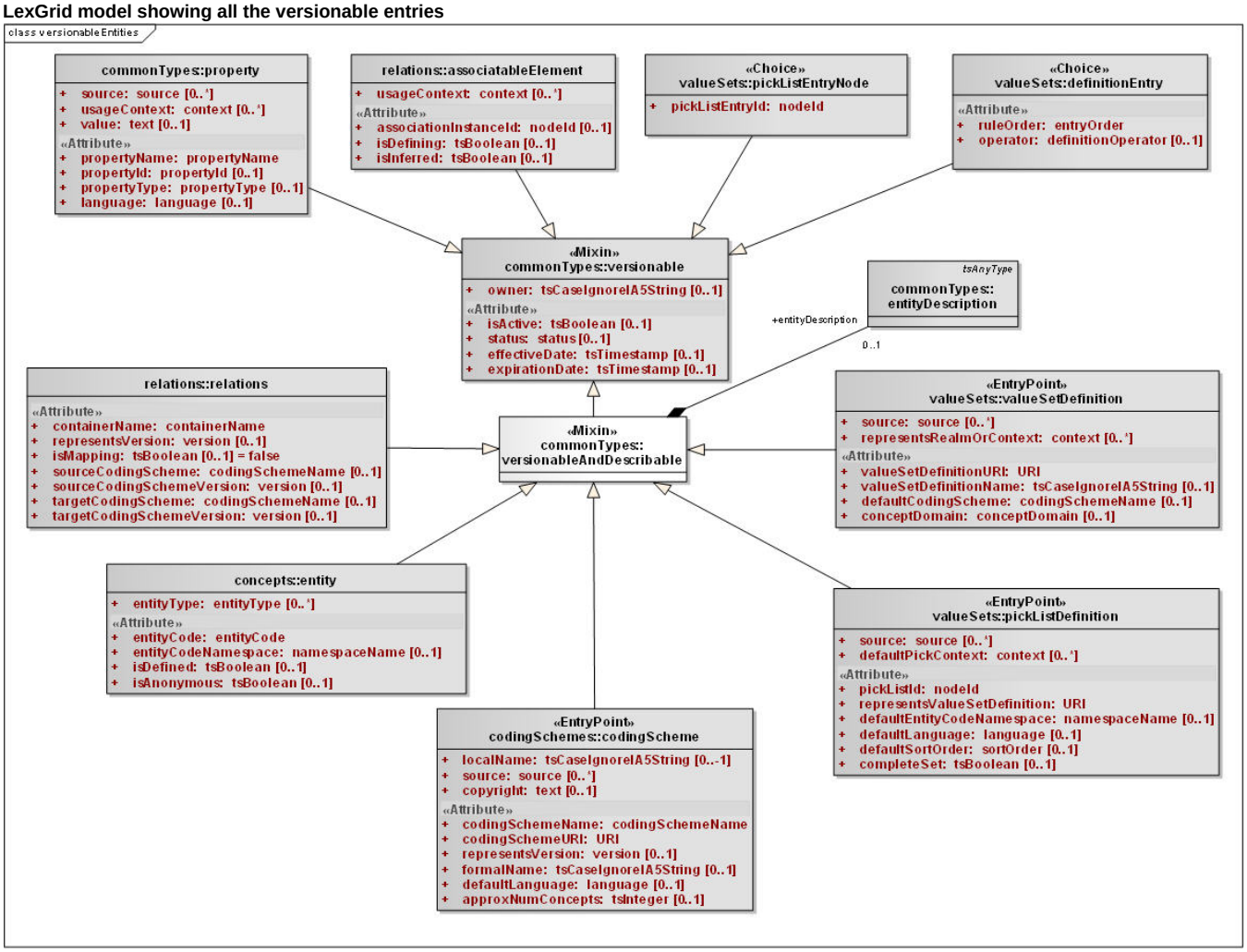
- **systemRelease** - A collection of coding schemes, value set definition, pick list definitions and/or revision records that are distributed as a unit.
- **editHistory** - An ordered collection of revisions.
- **revision** - An ordered collection of state changes that define the transformation of a set of resources from one consistent state to another.
    - changeAgent* - The local identifiers of the source that participated in this particular change. changeAgent must match a local id of a supportedSource in the corresponding mappings section.
    - **changeInstructions** - A human or machine readable set of instructions on how to apply this change.
    - **revisionId** - The unique identifier of this revision.
    - **editOrder** - The relative order that this revision is to be applied if in a systemRelease.
    - **revisionDate** - The end date for which this version is operative (considered committed).
- **Versionable** - A resource that can undergo change over time while maintaining its identity.
    - **Owner** - The owner of the resource. The specific semantics of owner is defined by the business rules of the implementer, including the rules of the owner field is absent.
    - **isActive** - True means that this resource is searchable and browsable if the temporal context of the operation falls between effectiveDate and expirationDate. False means that this resource is only accessible if requested by id or by a search that specifies that inactive retrievals are allowed. Default: True
    - **status** - The status code associated with the particular resource. The semantics and business rules of entryStatus are defined by the containing system, but there needs to be a mapping into isActive above.
    - **effectiveDate** - The date and time that this resource is considered to be active. To be considered active, isActive must be true, and the temporal context of the operation must be greater than effectiveDate. If omitted, all temporal contexts are considered to be valid.
    - **expirationDate** - The date and time that this resource is considered to become inActive. To be considered active, isActive must be true, and the temporal context of the operation must be less than expirationDate. If omitted, all temporal contexts are considered to be valid.
- **entryState** - Represents a change that occurred between the current state of the versionable entry and an immediately preceding state of the same entry.
    - **containingRevision** - The revision that contains this particular entry state change.
    - **relativeOrder** - The relative order that this state change should be applied within the context of the containing revision.
    - **changeType** - The type of change that occurred between this state and the previous.
        - **NEW** - Versionable entry is new in this revision. No previous state is available.
        - **MODIFY** - Entry has been modified between this state and the previous.
        - **VERSIONABLE** - Versionable attribute has changed since prior version. Versionable attributes include: isActive, status, owner, effectiveDate or expirationDate.

        - **DEPENDENT** - The status of a dependent entry has changed since the last version. Dependent entities include properties, codedEntries for codingSchemes, associationInstances, etc.
        - **REMOVE** - Versionable entry was removed as of this version. This is not the same as deprecated, as the entity ceases to exist in future versions.
    - **prevRevision** - The unique identifier of the state of this entry was at prior to this change.
- **changedEntry** - A top level versionable entry. Each changedEntry bucket can contain changedCodingSchemeEntry or changedValueSetDefinitionEntry or changedPickListDefinitionEntry.
- **changedCodingSchemeEntry** - this element is of type codingScheme and can contain changes to terminology entities.
- **changedValueSetDefinitionEntry** - this element is of type valueSetDefinition and can contain changes to the definition of a given value set.
- **changedPickListDefinitionEntry** - this element is of type pickListDefinition and can contain changes to the definition or the pickListEntrys.

LexGrid versionable entries

Authoring can be performed on all the versionable entries in LexGrid logical model.

LexGrid model showing all the versionable entries



LexGrid versionable entries description

- **codingScheme** - A resource that makes assertions about a collection of terminological entities.
- **entity** - A set of lexical assertions about the intended meaning of a particular entity code.
- **relations** - A collection of relations that represent a particular point of view or community.
- **associatableElement** - Information common to both the entity and data form of the "to" (or right hand) side of an association.
- **pickListDefinition** - An ordered list of entity codes and corresponding presentations drawn from a value set resolution
- **pickListEntryNode** - An inclusion (pickListEntry) or exclusion (pickListEntryExclusion) in a pick list definition.
- **valueSetDefinition** - A definition of a given value set. A value set definition can be a simple description with no associated value set entries, or it can consist of one or more definitionEntries that resolve to an enumerated list of entityCodes when applied to one or more codingScheme versions.
- **definitionEntry** - A reference to an entry code, a coding scheme or another value set definition along with the instructions about how the reference is applied. DefinitionEntrys are applied in entryOrder, with each successive entry either adding to or subtracting from the final set of entity codes.
- **property** - A description, definition, annotation or other attribute that serves to further define or identify a resource.

Authoring LexGrid non-versionable entries

LexGrid model entries like 'source', 'usageContext', etc which are not versionable, can also be modified but only in context with its parent entry. And also, any values to these attributes will be totally replacing the existing values.

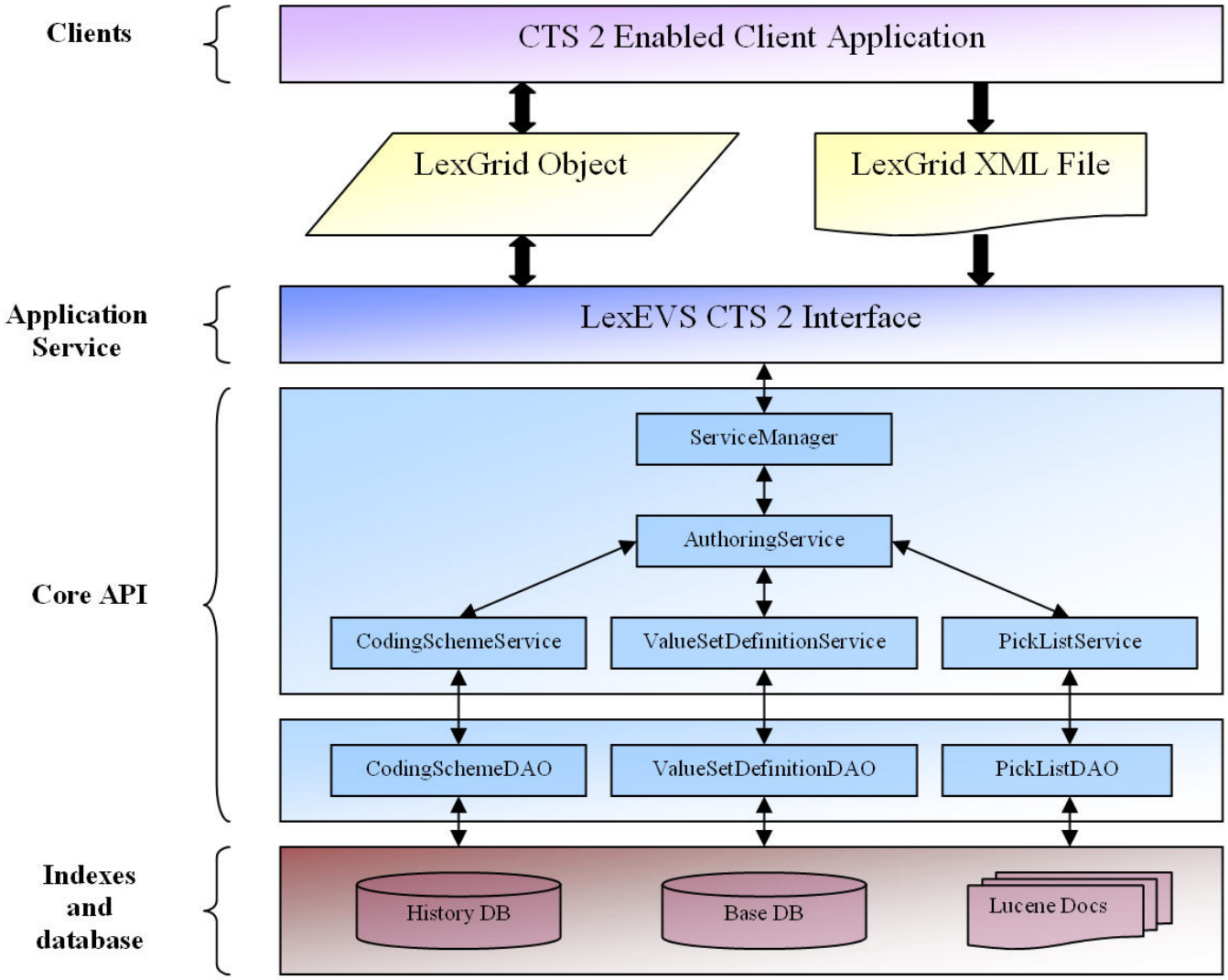
For example, if a coded concept 'abc' has a source 'company A, company B' and if the source 'company B' needs to be replaced by 'company C' but keep 'company A' as it is, we will need to pass in both source 'company A, company C' within concept 'abc' and have the changeType as 'MODIFY' for this concept.

LexEVS Authoring Architecture

LexEVS Authoring API can accept changes to the terminology contents only in the form of:

- LexGrid java objects, OR
- XML file in LexGrid format.

Diagram showing the architecture of LexEVS authoring environment



LexEVS Authoring process

Important: Before the changed terminology content is sent to LexEVS Authoring API to process, the client application should make sure that the contents are valid.

- The contents should either be in LexGrid XML format or a LexGrid java object.
- The entry level for the changed contents should be at either systemRelease or revision level.
- The input format will be validated against LexGrid schema for compliance. If validation fails, exception will be thrown.
- Each change request in the revision package is validated sequentially. Each individual request validation will be performed based on the premise that all requests that precede it have been applied (e.g. if there are two change requests, one to create a new coding scheme and the second to create a new entity for the coding scheme, the new entity request is validated on the assumption that the coding scheme has been created. The service may validate the entire revision or it may cease validation at any point after an error is detected - this is up to the service provider.

- If no errors are detected in previous step, the change package is submitted to the business rules "hook", which can apply additional validation, logging and error checking. If the business rules hook returns an error, no further processing is done and the result is returned to the submitter.
- If the submission passed step two, the complete set of changes will be applied to the service. If, for any reason, an error occurs, such as a network failure, a database failure, etc., the entire revision will be rolled back. Otherwise, the set of changes will be committed.

The intent of the change requests are as follows:

- **NEW** - to create a new versionable element
- **MODIFY** - to change the attributes of an existing versionable element
- **VERSIONABLE** - to change (or schedule a change of) the status of a versionable element within the context of the containing service.
- **REMOVE** - to remove a versionable element from the service. (Note that VERSIONABLE Retire should be used if the element and its history should remain)
- **DEPENDENT** - no changes are to be made to the named element itself, but a versionable element whose identity is dependent upon this element is to undergo a change.

#xample xml file containing changed coding scheme

```
<?xml version="1.0" encoding="UTF-8"?>
<ns3:systemRelease xmlns:ns1="http://LexGrid.org/schema/2010/01/LexGrid/valueDomains"
        xmlns:ns2="http://LexGrid.org/schema/2010/01/LexGrid/relations"
        xmlns:ns3="http://LexGrid.org/schema/2010/01/LexGrid/versions"
        xmlns:ns4="http://LexGrid.org/schema/2010/01/LexGrid/commonTypes"
        xmlns:ns5="http://LexGrid.org/schema/2010/01/LexGrid/codingSchemes"
        xmlns:ns6="http://LexGrid.org/schema/2010/01/LexGrid/concepts"
        xmlns:ns7="http://LexGrid.org/schema/2010/01/LexGrid/naming"
        xmlns:ns8="http://LexGrid.org/schema/2010/01/LexGrid/builtins"
        xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
        xsi:schemaLocation="http://LexGrid.org/schema/2010/01/LexGrid/versions http://LexGrid.org/schema/2010/01/LexGrid/versions.xsd"
        releaseId="Jan2010-Mayo" releaseURI="http://www.systemRelease.com/"
        releaseDate="2006-05-04T18:13:51" releaseAgency="Mayo">
        <ns4:entityDescription> </ns4:entityDescription>
        <ns3:editHistory>
                <ns3:revision revisionId="ModifiedGM" revisionDate="2006-01-01T18:13:51" editOrder="0">
                        <ns4:entityDescription>ModifiedGM</ns4:entityDescription>
                        <ns3:changeAgent>Mayo</ns3:changeAgent>
                        <ns3:changeInstructions dataType="String"> Instructions for the revision implementation.</ns3:changeInstructions>
                        <ns3:changedEntry>
                                <ns3:changedCodingSchemeEntry codingSchemeName=" Automobiles"
                                        codingSchemeURI="urn:oid:11.11.0.1" formalName="autos" defaultLanguage="en"
                                        approxNumConcepts="0" representsVersion="1.0" isActive="true" status="active"
                                        effectiveDate="2000-01-01T18:13:51" expirationDate="2020-01-01T18:13:51">
                                        <ns4:owner>Mayo</ns4:owner>
                                        <ns4:entryState containingRevision="ModifiedGM" relativeOrder="0"
                                                changeType="DEPENDENT"/>
                                        <ns4:entityDescription/>
                                        <ns5:mappings/>
                                        <ns5:properties/>
                                        <ns5:entities>
                                                <ns6:entity entityCode="C0001" entityCodeNamespace="Automobiles">
                                                        <ns4:entryState containingRevision="ModifiedGM" relativeOrder="3"
                                                                changeType="MODIFY"/>

                                                        <ns4:entityDescription>Powerful Car</ns4:entityDescription>
                                                        <ns6:entityType>concept</ns6:entityType>
                                                        <ns6:presentation propertyName="textualPresentation" propertyId="c1"
                                                                propertyType="presentation" language="en" isPreferred="true"
                                                                isActive="true">
                                                                <ns4:entryState containingRevision="ModifiedGM" relativeOrder="0"
                                                                        changeType="MODIFY"/>
                                                                <ns4:value dataType="C0001-Format">Powerful Car</ns4:value>
                                                        </ns6:presentation>
                                                </ns6:entity>
                                        </ns5:entities>
                                </ns3:changedCodingSchemeEntry>
                        </ns3:changedEntry>
                </ns3:revision>
        </ns3:editHistory>
</ns3:systemRelease>
```

Querying data based on version information

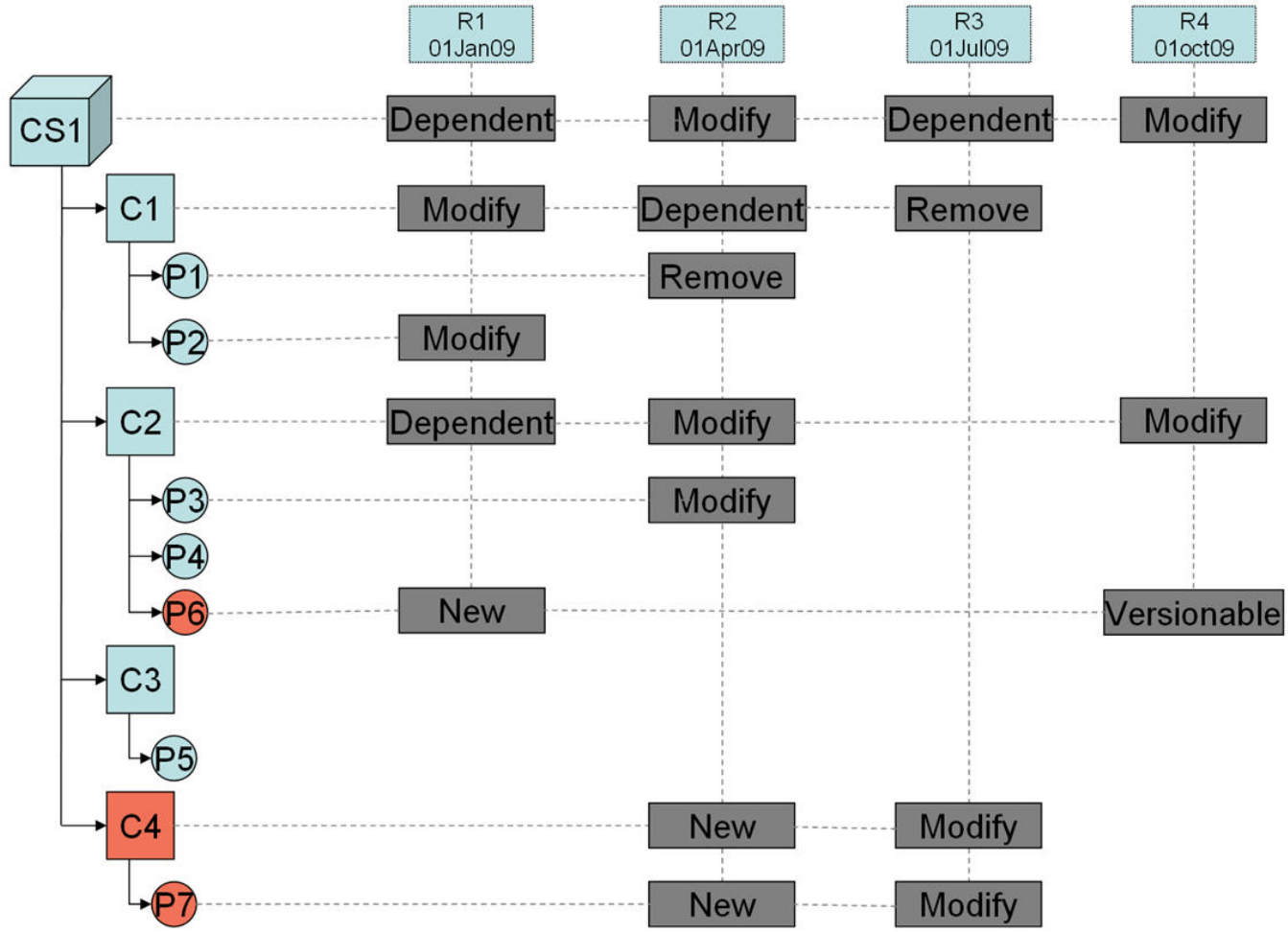
LexEVS Authoring API provides ability to query terminology content based on:

- revision identifier
- specific date and time

Query data based on revision identifier

This allows users to get the state of an entity like concept code, coding scheme, pick list, etc, at the given revision.
Exception will be thrown if the revision identifier was not found in the system.
And if that entity (ex concept code) had no changes for that particular revision, the API will return the state of that entity when that revision was applied to the system.

Example of query based on revision identifier

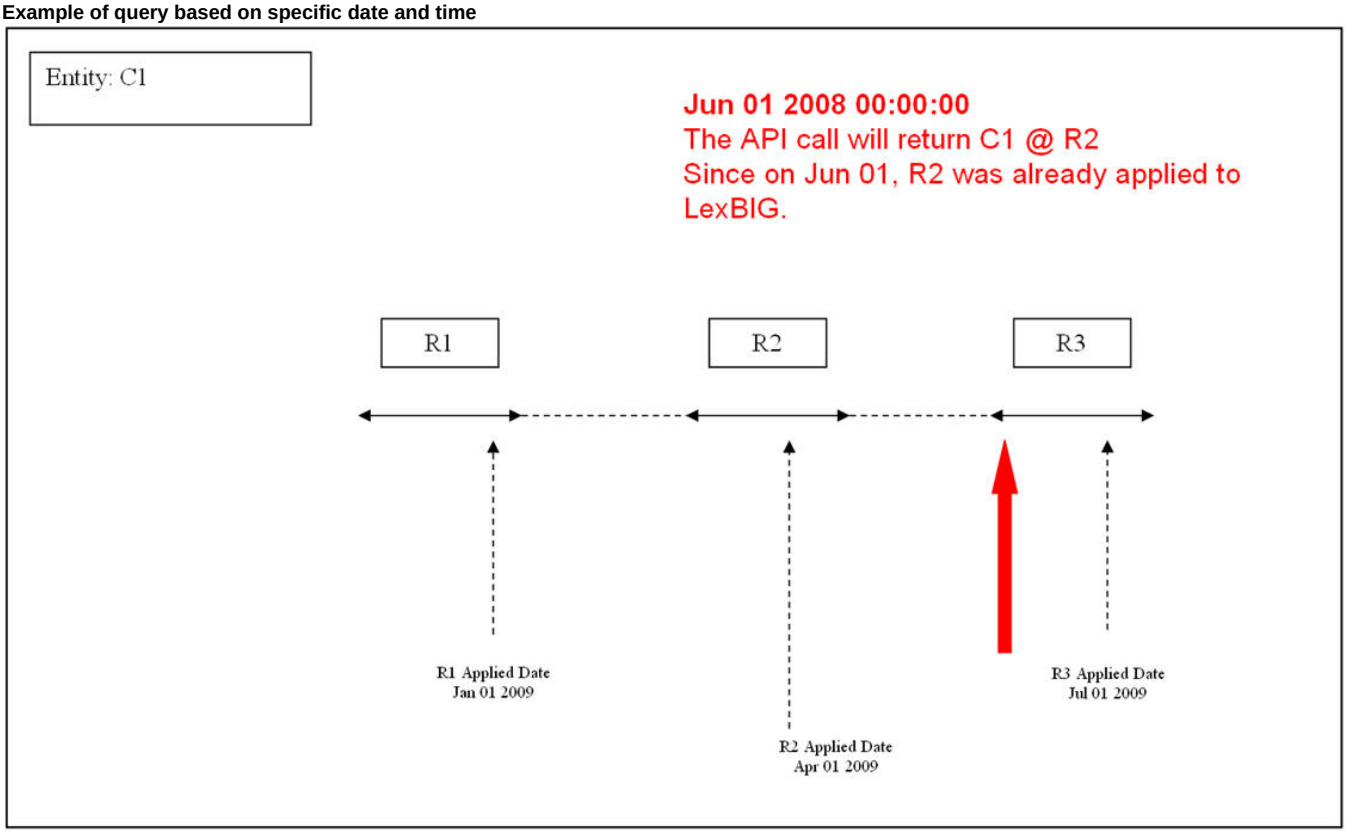


In the above example, coding scheme 'CS1' is revised in 4 different instances R1, R2, R3 and R4 at respective dates. The query API will be able to provide the states of the versionable elements at different revisions. Example, API will be able to return the state of concept 'C1' at revision R2.

Query data based on specific date and time

This allows users to get the state of an entity like concept code, coding scheme, pick list etc, at the give date and time.
Exception will be thrown if no data was found for given date and time.

Example of query based on specific date and time



CTS 2 Authoring profile:

The CTS2 SFM calls for the ability to create, maintain and update code systems, concepts, and associations as separate entities. The LexGrid and LexEVS model views all three of them as aspects of code systems, and its incremental revision approach allows any or all of them to be changed as a single unit. The LexEVS model also subsumes the notion of a "code system supplement", as a collection of one or more revisions to a code system can be packaged as a "system release", with its own provenance, activation dates, etc., and can be applied to external code systems independently.

The Terminology Authoring Profile is intended to provide the capability to robustly query and access terminology content, as well as directly modify the terminology content. This includes the ability to modify code system content, value set content, as well as the metadata pertaining to each. This profile includes the functions necessary to administer and search terminology content as outlined in the CTS2 Query Profile as well as the Terminology Administration Profile.

List of Authoring profiles as in CTS2 SFM

CTS2 Function	CTS2 Function Description	LexEVS Implementation
Create Code System	Create a new Code System to contain a set of new coded concepts. The Code System is created by defining the set of meta-data properties that describe it.	LexEVS provides ability to load Code System from various source format. Here are the loaders available: • HL7 RIM DB • LexGrid Text file • LexGrid XML file • NCI MetaThesaurus • OBO • OWL • RedLex Protégé Frames • UMLS Semantic Net • UMLS Source From RRF files • UMLS Source From SQL
Maintain Code System Version	Update Code System meta-data properties.	The LexEVS Authoring services will provide ability to update meta-data of a Code System as well as all its entities.
Update Code System Version Status	Changes the status of a code system version (suspended, reinstated, canceled, removed).	The LexEVS Authoring services will provide ability to modify the status of a Code System version.

Create Code System Supplement	Create a new Code System Supplement as a container of a set of concepts and concept properties to be appended to a target code system. Does not add the concepts and properties.	LexEVS treats Code System Supplement as any other Code System. The URI of target Code System will be included in SupportedCodingScheme and imported flag will be set to 'true'. All the contents of target/base Code System will be imported into this Supplement Code System. The rest of the function will be similar to **'Create Code System'** described above
Maintain Code System Supplement	Update Code System Supplement meta-data properties and add concepts and properties to code system.	The LexEVS, Code System Supplement is treated like any other Code System, thus provides an ability to update its meta-data properties, add or modify concepts and its properties.
Create Concept	Create concept to be included in a Code System. The new concept is defined by the set of meta-data properties that describe it, which may include its proper placement via association binding within the hierarchy of the Code System.	The LexEVS Authoring services will provide ability to add and update entities in a Code System.
Maintain Concept	Update Concept meta-data properties.	The LexEVS Authoring services will provide ability to add and update entities in a Code System.
Update Concept Status	Changes the status of a code system concept (suspend, reinstate, cancel, remove).	The LexEVS Authoring services will provide ability to add and update entities in a Code System.
Create Association Type	Create a new relationship type (as intended by the association type class of the conceptual model), an instance of which may be used to link two concepts. A list of code system IDs can be supplied if the intent is to restrict use to specific code systems. The default is availability to all code systems present on the server.	The LexEVS Authoring services will provide ability to create and maintain entities of type Association in a Code System.
Maintain Association Type	Update or deprecate an Association type that may be used to link two concepts.	The LexEVS Authoring services will provide ability to create and maintain entities of type Association in a Code System.
Create Value Set	Create a Value Set (extensional or intensional) that is defined by a computable expression that can be resolved to an exact list of coded concepts at any given point in time.	The LexEVS Authoring services will provide ability to create and maintain value set definitions.
Maintain Value Set	Update properties or expression of a value set definition (extensional and intensional value sets).	The LexEVS Authoring services will provide ability to create and maintain value set definitions.
Update Value Set Status	Changes the status of a value set version (suspend, reinstate, cancel, remove).	The LexEVS Authoring services will provide ability to create and maintain value set definitions.
Create Concept Domain	Create a Concept Domain.	In LexEVS, a coding scheme can contain entities of type 'Concept Domain'. So similar to function 'Create Concept', an entity of type 'concept domain' can be added and maintained in a coding scheme. And similar to an entity of type 'concept', concept domain can have properties, participate in associations etc.
Maintain Concept Domain	Update properties of a Concept Domain, including bindings to value sets within usage contexts	Similar to 'maintain concept' function described above, in LexEVS, a coding scheme can contain entities of type 'Concept Domain' and it can be added or modified (including its properties) in a coding scheme. Any changes to the value set binding should be performed separately using 'maintain value set' function.
Create Usage Context	Create a Usage Context.	Similar to 'create concept domain' function described above, in LexEVS, a coding scheme can contain entities of type 'Usage Context' and it can be added or modified in a coding scheme.
Maintain Usage Context	Update properties of a Usage Context	Similar to 'maintain concept' function described above, in LexEVS, a coding scheme can contain entities of type 'Usage Context' and it can be added or modified (include its properties) in a coding scheme.
Update Association Status	Update the status of an association (active, inactive, cancelled etc). This allows a Terminology User to activate or inactivate a given association, thus changing its availability for access by other terminology service functions	LexEVS Authoring API provides ability to change the status of an association (a triple).
Create Association	Relates a single specific coded concept within a specified code system (source) to a corresponding single specific coded concept (target) within the same or another code system, including identification of a specified Association type.	LexEVS Authoring API provides ability to create and maintain association between coded concepts from one or more code system.

Create Lexical Association between Coded Concept	Relates a set of one or more coded concepts within a specified code system (source)to a corresponding set of one or more coded concepts (target) within that system or another code system using a set of lexical rules (matching algorithms) to generate the Association. The "Source Search Criteria" allows for identification of a subset of the Source Code System to apply the matching algorithm to, if required (this may include limiting the version of the code system).	This functionality is out of scope for this release.
Create Rule Based Association between Coded Concept	Relates a set of zero or more coded concepts within a specified code system (source)to a corresponding set of zero or more coded concepts (target) within that system or another code system using a set of description logic or inference rules that either assert or infer Associations. The "Source Search Criteria" allows for identification of a subset of the Source Code System to apply the matching algorithm too, if required (this may include limiting the version of the code system).	This functionality is out of scope for this release.