LexEVS 6.0 Design Document - Solution Architecture

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

Solution Architecture

Proposed technical solution to satisfy the following requirements:

- Provide support for Value Sets.
- Develop within LexEVS the ability to provide local extensions to code sets and maps among code sets.
- Develop within LexEVS other capabilities called for in the CTS2 Specification.

Required CTS 2 Functionality

The required LexEVS functionality to support CTS2 addresses several broad categories.

Administrative Operations

Import Operations The CTS2 SFM calls for the ability to import code systems, code system revisions, value set versions and association versions. The current LexEVS model does not differentiate between the importation of a complete code system and an incremental update in the form of a code system revision, but the functionality is sufficient to fully meet the requirements of both operations. Note, however, that the incremental update functionality of LexEVS has not been fully implemented as of version 5.1, and will be completed in order to meet these requirements. The import of association versions will be also be absorbed as part of the code system revision functionality. The current LexEVS implementation already supports the import of value domain definitions, although incremental updates have not been implemented and will be provided.

Export Operations The CTS2 SFM calls for the ability to export code systems, associations and value sets. The current LexEVS implementation supports the ability to export complete code systems in LexGrid XML and OBO formats. At the moment, it does not support the ability to export value domain definitions or pick lists which will be provided. The CTS2 SFM also calls for the ability to provide filter criteria in the exports although the use and functions of such filters are not totally clear.

While the SFM does not spell a minimal set of export requirements, we believe that it will be necessary to support LexGrid XML and RDF / OWL. We also believe that there are use cases that will require the ability to export a set of changes as a "delta" from a previous version - a set of changes that can be applied to another image that will supply the appropriate update.

Code System Status Changes The current LexEVS implementation already supports a superset of this functionality.

Notification While the need to support notification has been anticipated in the current LexGrid architecture, it has not been completely modeled or implemented. Analysis, however, has identified a set of requirements that extend beyond the basic ones identified in the CTS 2 SFM. As an example, a use case was identified where an administrator needed to be notified when the contents of a concept that was referenced by a value set changed.

Architectures and corresponding implementations for notification and event generation already exist. While it will be necessary to tie these events in to the LexEVS implementation, we do not plan to implement any of the notification tooling directly but, instead will implement it in such a way that it can be tied into a standards compliant event and messaging architecture.

Search and Query Operations

Code Systems LexEVS already implements a superset of the code system search and access requirements with the exception that search criteria and the "query control" aspects are combined as a single operation.

Value Sets The CTS2 SFM calls for the ability to list value sets, return value set details and list value set contents. It also calls for a determination of value set subsumption and queries about concept membership. The current LexEVS implementation supports all of these functions with the exception of value set subsumption. It should be noted, however, that there are two possible interpretations of "subsumption" - an extensional and intensional subsumption. Testing extensional subsumption determines whether one value set subsumes another based on their current resolutions. Testing extensional subsumption, however, is more difficult, as it involves the determination whether subsumption is necessary. We intend to postpone subsumption pending further clarification of the use case.

Concept Domains and Usage Contexts The HL7 notion of "Concept Domain" was originally architected to align with the ISO 11179 Enumerated Conceptual Domain. It has since evolved, however, to be a more abstract entity that, if anything, is closer to the ISO 11179 Data Element Concept. The CTS SFM shows the role of Concept Domain as coupling a value set with a set of designations. While LexEVS supports this particular functionality through pick list implementation, we are not certain that this model will meet all of HL7's needs, as HL7 Edition 3 views concept domains as controlling the coupling of data elements with particular enumerated conceptual domains. We intend to model concept domain, usage context and jurisdictional domain as code system entities instead of making them first class model elements. The concept domain binding and concept to concept domain functionality is partially implemented in the LexEVS pick list model, but additional modeling and development will be done to provide the full functionality required by the SFM.

Association Related Queries The CTS2 SFM calls for the ability to enumerate associations, compute the transitive path between two concept codes, determine whether one coded attribute is subsumed by another and return the details of an association. Some of the requirements are a bit unclear on whether they are calling for the ability to query association types or the actual set of associations (relation) coupled with the type. LexEVS provides a rich set of functionality to perform the latter and provides all of what we believe to be necessary to support the former.

Full subsumption queries imply the use of a reasoner. We see this as a non-trivial task, as the different terminologies are based on different types of description logics and, even within the same family of description logic there are different algorithms that can produce different results based on completeness requirements. The LexEVS package does not currently support (a) reasoners and (b) the ability to supply a pre-coordinated expression as either the input or output of a function. We will postpone the implementation of compositional expression pending the clarification of whether LexEVS can or should support formal reasoning.

Authoring and Curation Operations

"Authoring" and "incremental update" are closely related notions, but there is an important distinction between them. Incremental update, as specified in the current version of the LexEVS model, assumes that any set of changes will transform the underlying entity (code system, value domain, pick list) from one consistent state to another. Authoring, however, requires an additional ability to save entities in states that are neither valid nor complete. 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.

It is assumed 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.

Code System Authoring and Curation 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.

High Level Architecture

Structure of the CTS 2 Service

The CTS 2 specification defines several functional profiles which are a focused subset of the functionality of a CTS 2 implementation. Functional profiles are defined to subset a group of operations which must be supported in order to claim conformance to the profile.

The following functional profiles are considered in scope for LexEVS 6.0:

CTS 2 Query Profile

- Searching and querying terminologies
- Provide access to terminology content and representational structures (description logic) consistent with the terminology author's intent.

Terminology Administration Profile

- Restricting administrative access
- Obtaining and loading terminologies

- Maintaining terminology access
- Control Content Access

Terminology Authoring Profile

- Functional terminology analysis/query
- Direct terminology edits

Each profile specifies the minimal functional coverage as represented in the following tables.

CTS 2 Query Profile

Function	Description
List Code Systems	The ability to provide a listing of the available code systems that meet input search criteria.
Return Code System Details	The ability to retrieve a specific code system attributes (synonyms, associations) and other metadata.
List Code System Concepts	The ability to retrieve a list of all of the concepts, with associated attributes (synonyms, associations) and other metadata that meet input criteria.
Return Concept Details	The ability to retrieve a specific concept, with associated attributes (synonyms, associations) and other metadata.
List Value Sets	The ability to determine what value sets are available to a Terminology Service. This includes seeing a listing of the available value sets that match some search criteria, as well as the details pertaining to each value set available to the terminology service.
Return Value Set Details	The ability to retrieve a specific value set, with associated attributes and other metadata.
List Value Set Contents	The ability to see a listing of specific concepts, as well as the details pertaining to each concept in any of the given value sets available to a terminology service.
Check Concept Value Set Membership	The ability to validate that a given concept exists in a given value set.
List Concept Domains	The ability to determine what concept domains are available to a Terminology Service.
Return Concept Domain Details	The ability to retrieve a specific concept domain, with associated attributes and other metadata.
List Concept Domain Bindings	The ability to see a listing of specific value sets that are bound to a concept domain in specified usage contexts.
Check Concept Domain Membership	The ability to validate that a given concept code is bound to a given concept domain.
List Usage Contexts	The ability to determine what usage contexts are available to a Terminology Service.
Return Usage Context Details	The ability to retrieve a specific usage context, with associated attributes and other metadata.
List Associations	The ability to determine what associations are available on the terminology service by browsing a list of available associations on the CTS 2 instance that meet specified search criteria.
Return Association Details	The ability to retrieve metadata on available associations in the CTS 2 service instance.
List Association Types	Returns the details for the known attributes (metadata) of a coded concept
Return Association Type Details	The ability to return all information for a Association type.
Check Value Set Subsumption	Determine whether one of the two supplied value sets subsumes the other
Check Concept to Concept Domain Association	Determine whether the supplied coded concept exists in a code system in use for the specified concept domain, optionally within specific usage contexts.
Determine Transitive Concept Relationship	Determine whether there exists a transitive relationship between two concepts
Compute Subsumption Relationship	Determine Whether One Concept Subsumes a Second

Terminology Administration Profile

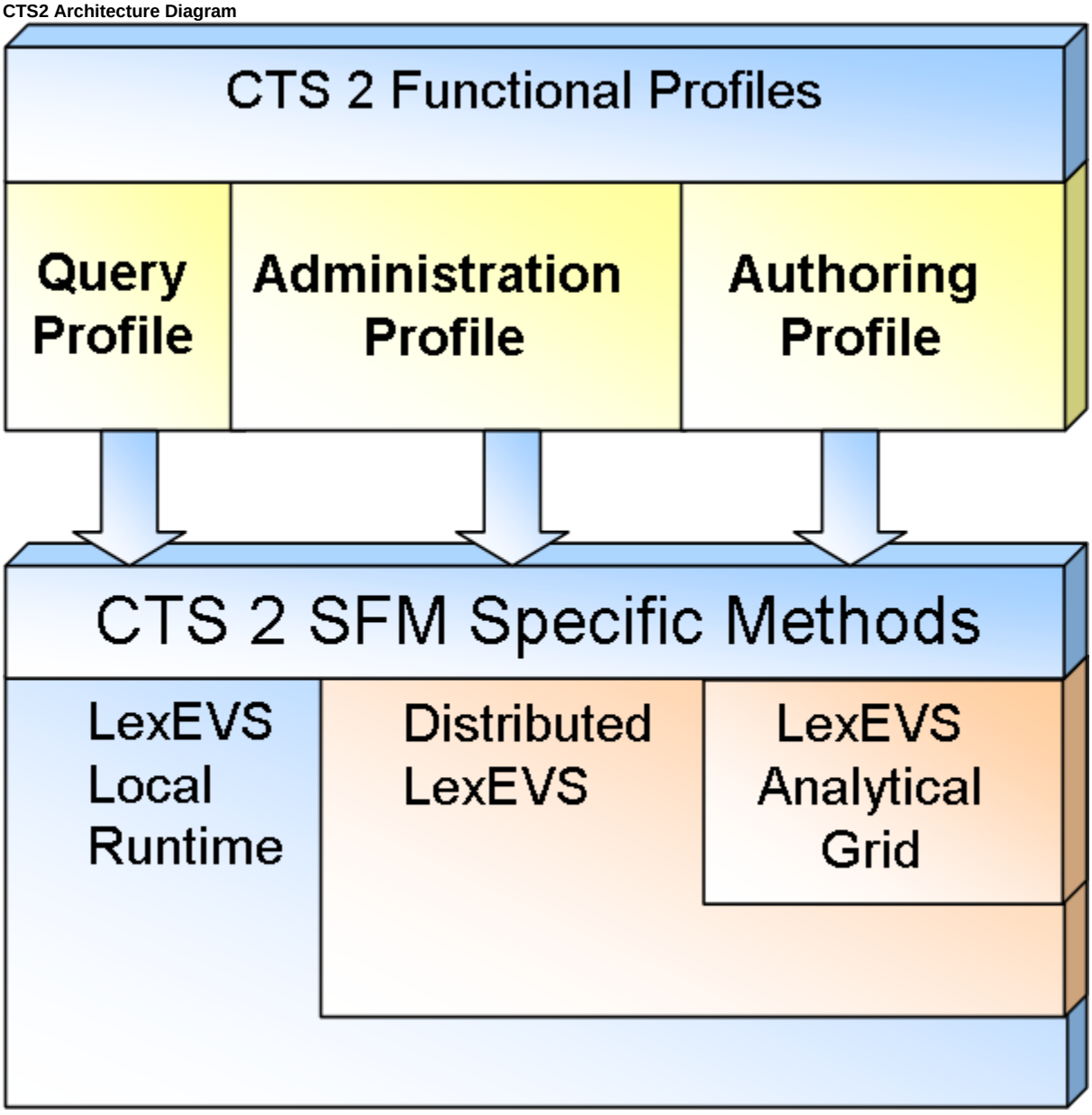
Function	Description
Import Code System	Terminology content would be loaded into the terminology server as an entire terminology load or skeleton load (i.e. load of structure without loading the nodes).
Import Code System Revision	Terminology content would be loaded into the terminology server as a delta or set of changes from the previous version of the terminology.
Import Value Set Version	Ability to import values sets
Import Association version	Ability to import Associations
Export Association	Ability to export Association Type instances
Export Code System Content	Terminology content would be exported either in whole or in part based on filtering against terminology properties. The export format may also be specified.
Change Code System Status	Terminology content status would be changed, thus changing its availability for access by other terminology service functions.
Register for Notification	A client registers for notification so that an electronic notification would be sent to subscribed users in the event of a change to the specified terminology element.
Update Notification Registration	Subscription notification information can be updated for a subscriber's notification account.
Update Notification Registration Status	Updates the status of a notification registration.

Terminology Authoring Profile

Function	Description
Create Code System	The ability to 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.
Maintain Code System Version	The ability to maintain the content and metadata of a version for a code system.
Update Code System Version Status	The ability to modify the status of a code system.
Create Concept	The ability to define and add a new concept to a code system.
Maintain Concept	The ability to modify a concept that exists in a code system.
Update Concept Status	The ability to modify the status of a concept that exists in a code system.
Create Value Set	The ability to create a dynamic value set that is defined by a computable expression that can be resolved to an exact list of coded concepts at any given point in time.
Maintain Value Set	Update properties or expression of a value set definition (extensional and intensional value sets).
Update Value Set Status	The ability to modify the status of a value set.
Create Concept Domain	The ability to define and add a new concept domain.
Maintain Concept Domain	The ability to modify a concept domain, including bindings to value sets within usage contexts.
Create Usage Context	The ability to define and add a new usage context.
Maintain Usage Context	The ability to modify a usage context.
Terminology Administration Profile	The Terminology Administration profile is intended to provide the functional operations necessary for terminology administrators to be able to access and make available terminology content obtained from a Terminology Provider.
Create Association	The ability to create an association between concepts.
Update Association Status	The ability to update the status of an association between concepts.
Create Association Type	The ability to create a new Association type that may be used to link two concepts.

Maintain Association Type	The ability to modify or deprecate an existing Association type that may be used to link two concepts.
Create Lexical Association Between Coded Concepts (optional for this profile)	The ability to instantiate an association between two sets of coded concepts using a set of lexical rules (matching algorithms) to generate the associations .
Create Rules Based Association Between Coded Concepts (optional for this profile)	The ability to instantiate an association between two sets of coded concepts using a set of description logic or inference rules that either assert or infer mappings between two Code Systems.
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
Maintain Code System Supplement	Update Code System Supplement meta-data properties and add concepts and properties to code system

CTS2 Architecture Diagram



Semantic Profiles

Semantic profiles identify a named set of metamodels that are to be supported by the operations specified in the functional profiles.

The following semantic profile is considered in scope for LexEVS 6.0:

Mature Terminology Profile

- Best practices conformance for the terminology

- Terminologies in the Mature Terminology Profile make an attempt to conform to many of terminology best practices that are, for example, outlined in *Desiderata for Controlled Medical Vocabularies in the Twenty-First Century, James J. Cimino*.
- Sample Terminologies include: SNOMED CT, ICD 10 CM, LOINC, RxNorm, NDF / NDF-RT

This profile best fits the existing NCI terminologies.

Conformance Profiles

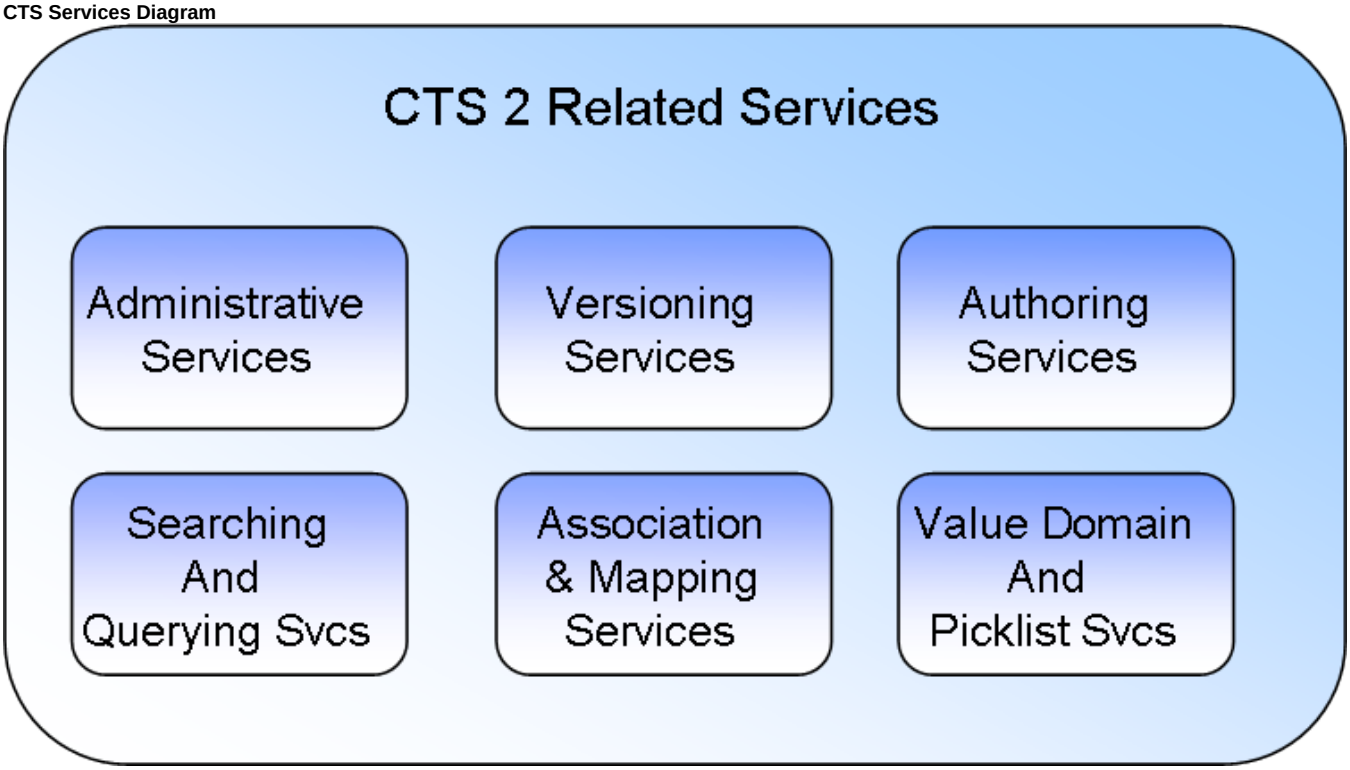
Conformance profiles are intended to focus specific implementations to address a specific class of functionality and minimum trait sets for each functional class. LexEVS 6.0 intends to implement to the following conformance:

Profile	Mature Terminology Semantic Profile
CTS 2 Query Functional Profile	CTS2 Query - Mature Terminology Conformance Profile
Terminology Administration Functional Profile	Terminology Administration - Mature Terminology Conformance Profile
Terminology Authoring Functional Profile	Terminology Authoring - Mature Terminology Conformance Profile

Sub-Categorization of CTS 2 Services

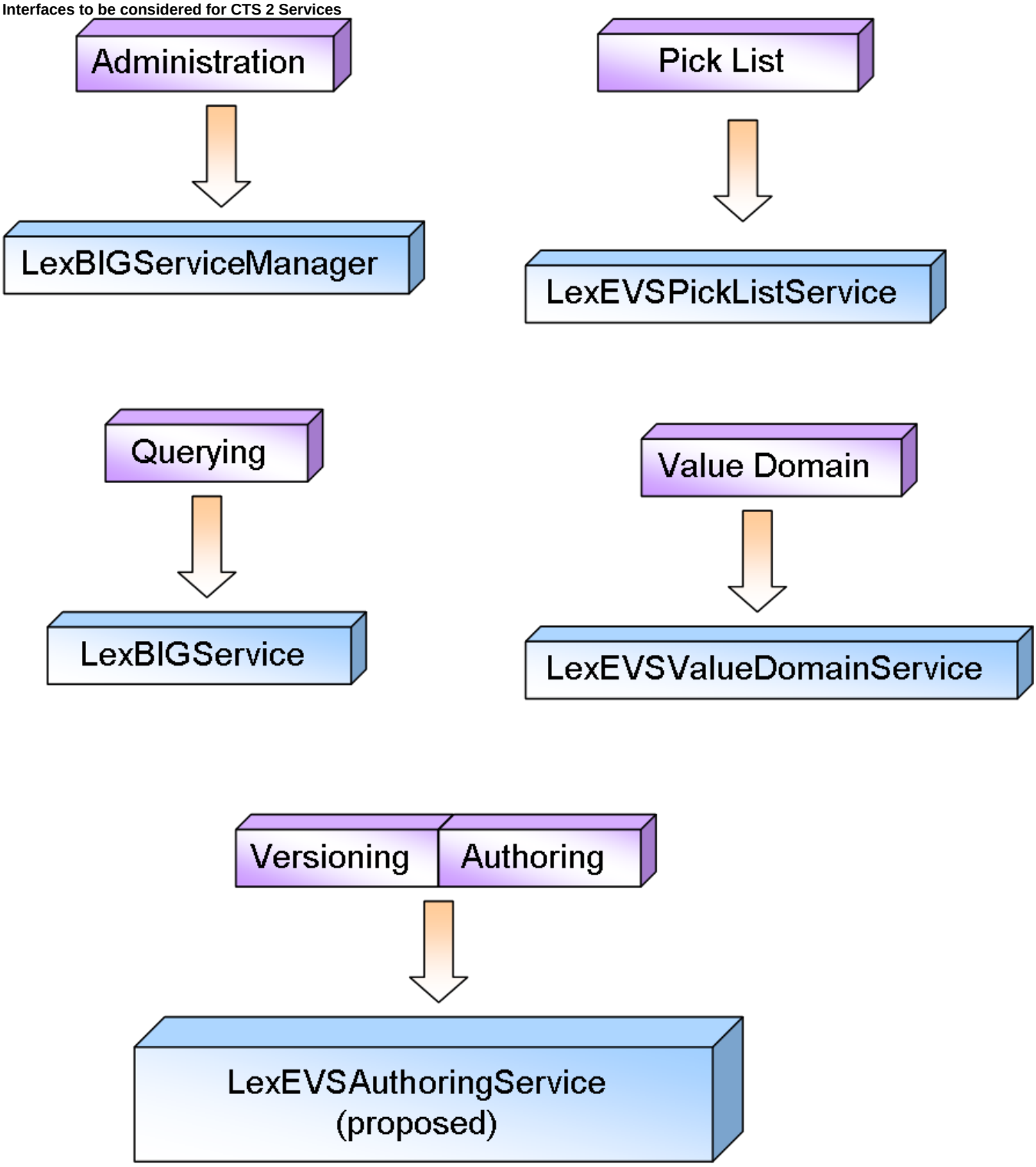
CTS 2 Services can be further categorized from the above profile details.

CTS Services Diagram



Service Interfaces for CTS 2

Interfaces to be considered for CTS 2 Services



High Level Design Diagram

The LexEVS 6.0 infrastructure exhibits an n-tiered architecture with client interfaces, server components, domain objects, data sources, and back-end systems (architecture diagram). This n-tiered system divides tasks or requests among different servers and data stores. This isolates the client from the details of where and how data is retrieved from different data stores.

The system also performs common tasks such as logging and provides a level of security for protected content. Clients (browsers, applications) receive information through designated application programming interfaces (APIs). Java applications communicate with back-end objects via domain objects packaged within the client.jar. Non-Java applications can communicate via SOAP (Simple Object Access Protocol) or REST (Representational State Transfer) services.

Most of the LexEVS API infrastructure is written in the Java programming language and leverages reusable, third-party components. The service infrastructure is composed of the following layers:

Application Service Layer - accepts incoming requests from all public interfaces and translates them, as required, to Java calls in terms of the native LexEVS API. Non-SDK queries are invoked against the Distributed LexEVS API, which handles client authentication and acts as proxy to invoke the equivalent function against the LexEVS core Java API. The caGrid and SDK-generated services are optionally run in an application server separate from the Distributed LexEVS API.

The LexEVS caCORE SDK services work directly against the database, via Hibernate bindings, to resolve stored objects without intermediate translation of calls in terms of the LexEVS API. However, the LexEVS SDK services do still require access to metadata and security information stored by the Distributed and Core LexEVS API environment to resolve the specific database location for requested objects and to verify access to protected resources, respectively.

From the client prospective, the LexEVS services will function as "ports" accessible through the caGrid 1.3 service architectural model. LexEVS services will follow the caGrid architecture for analytical and data services. See the caGrid 1.3 documentation for architectural details: https://cabig.nci.nih.gov/workspaces/Architecture/caGrid/

Core API Layer - underpins all LexEVS API requests. Search of pre-populated Lucene index files is used to evaluate query results before incurring cost of database access. Access to the LexGrid database is performed as required to populate returned objects using pooled connections.

Data Source Layer - is responsible for storage and access to all data required to represent the objects returned through API invocation.

High Level Design Diagram