Federated Content Search for Lexical Resources (LexFCS)

Specification

Erik Körner, Thomas Eckart, Uwe Kretschmer, Axel Herold, Frank Wiegand, Frank Michaelis, Claus Zinn, Matthias Bremm, Louis Cotgrove, Thorsten Trippel, Felix Rau, Anne Klee, Daniel Werning, Dominik Blöse

Version 0.3, 2025-06-20

Table of Contents

Chapter 1. Introduction

The *Lexical Search for Federated Content Search (LexFCS)* specification is an extension of the *CLARIN Federated Content Search (CLARIN-FCS) - Core 2.0* specification that allows search and retrieval of *lexical resources* including dictionaries, encyclopedias, normative data, terminological databases, ontologies etc.

1.1. Terminology

The key words `MUST`, `MUST NOT`, `REQUIRED`, `SHALL`, `SHALL NOT`, `SHOULD`, `SHOULD NOT`, `RECOMMENDED`, `MAY`, and `OPTIONAL` in this document are to be interpreted as in RFC2119.

1.2. Glossary

> **NOTE** Based on *Glossary* in FCS Core 2.0 specification.

CLARIN-FCS, FCS
: CLARIN federated content search, an interface specification to allow searching within resource content of repositories.

Client
: A software component, which implements the interface specification to query Endpoints, i.e. an aggregator or a user interface.

CQL
: Contextual Query Language, previously known as Common Query Language, is a domain-specific language for representing queries to information retrieval systems, such as search engines, bibliographic catalogs and museum collection databases.

Data View
: A Data View is a mechanism to support different representations of search results, e.g. a "hits with highlights" view, an image, or a geolocation.

Endpoint
: A software component, which implements the CLARIN-FCS interface specification, and translates between CLARIN-FCS and a search engine.

Hit
: Data or a subdivision of data returned by a Search Engine that matches the search criterion. What is considered a Hit highly depends on the Search Engine.

Interface Specification
: Common harmonized interface and suite of protocols that repositories are required to implement.

Resource

A searchable and addressable entity offered by an Endpoint, such as a text corpus or a multi-modal corpus.

Result Set

An (ordered) set of Hits that match a search criterion produced by a search engine as the result of processing a query.

Search Engine

A software component within a repository that allows for searching within the repository contents.

SRU

Search and Retrieve via URL is a protocol for Internet search queries. Originally introduced by Library of Congress LOC-SRU12, later standardization process moved to OASIS OASIS-SRU12, OASIS-SRU20.

1.3. Normative References

NOTE Based on *Normative References* in FCS Core 2.0 specification.

RFC2119

Key words for use in RFCs to Indicate Requirement Levels, IETF RFC 2119, March 1997, https://www.ietf.org/rfc/rfc2119.html

XML-Namespaces

Namespaces in XML 1.0 (Third Edition), W3C, 8 December 2009, http://www.w3.org/TR/2009/REC-xml-names-20091208/

OASIS-SRU-Overview

searchRetrieve: Part 0. Overview Version 1.0, OASIS, January 2013, http://docs.oasis-open.org/search-ws/searchRetrieve/v1.0/os/part0-overview/searchRetrieve-v1.0-os-part0-overview.html (DOC), (PDF)

OASIS-SRU12

searchRetrieve: Part 2. SRU searchRetrieve Operation: APD Binding for SRU 1.2 Version 1.0, OASIS, January 2013, http://docs.oasis-open.org/search-ws/searchRetrieve/v1.0/os/part2-sru1.2/searchRetrieve-v1.0-os-part2-sru1.2.html (DOC), (PDF)

OASIS-SRU20

searchRetrieve: Part 3. SRU searchRetrieve Operation: APD Binding for SRU 2.0 Version 1.0, OASIS, January 2013, http://docs.oasis-open.org/search-ws/searchRetrieve/v1.0/os/part3-sru2.0/searchRetrieve-v1.0-os-part3-sru2.0.html (DOC), (PDF)

OASIS-CQL

searchRetrieve: Part 5. CQL: The Contextual Query Language version 1.0, OASIS, January 2013, http://docs.oasis-open.org/search-ws/searchRetrieve/v1.0/os/part5-cql/searchRetrieve-v1.0-os-

part5-cql.html (DOC), (PDF)

LOC-SRU12

SRU Version 1.2: SRU Search/Retrieve Operation, Library of Congress, http://www.loc.gov/standards/sru/sru-1-2.html

LOC-CQL

The *Contextual Query Language*, Library of Congress, https://www.loc.gov/standards/sru/cql/, see also OASIS-CQL

LOC-CQLCS

The *CQL Context Set*, Library of Congress, https://www.loc.gov/standards/sru/cql/contextSets/theCqlContextSet.html

CLARIN-FCSCore20

CLARIN Federated Content Search (CLARIN-FCS) - Core 2.0 specification, SCCTC FCS Task-Force, https://office.clarin.eu/v/CE-2017-1046-FCS-Specification-v89.pdf, (Sources), (HTML Build), (PDF Build),

1.4. Non-Normative References

UD

Universal Dependencies project, that provides terminology for linguistic tags, features and relations, https://universaldependencies.org/

EDTF

Extended Date/Time Format (EDTF) Specification, February 4, 2019, The Library of Congress, https://www.loc.gov/standards/datetime/

1.5. Typographic and XML Namespace conventions

Sections that are still in discussion and not yet finalized are marked with `(WIP)` and may optionally have some *NOTE* admonition blocks. Details and specifications `SHOULD NOT` be considered stable.

The following typographic conventions for XML fragments will be used throughout this specification:

- `<prefix:Element>`

  An XML element with the Generic Identifier *Element* that is bound to an XML namespace denoted by the prefix *prefix*.

- `@attr`

  An XML attribute with the name *attr*.

- `string`

  The literal *string* `MUST` be used either as the content of an element or value of an attribute.

Endpoints and Clients MUST adhere to the XML-Namespaces specification. The CLARIN-FCS interface specification generally does not dictate whether XML elements should be serialized in their prefixed or non-prefixed syntax, but Endpoints MUST ensure that the correct XML namespace is used for elements and that XML namespaces are declared correctly. Clients MUST be agnostic regarding syntax for serializing the XML elements, i.e. if the prefixed or un-prefixed variant was used, and SHOULD operate solely on *expanded names*, i.e. pairs of *namespace name* and *local name*.

For a list of common XML namespace names and prefixes see the table "XML Namespaces and prefixes" in Section 1.5 of the FCS Core 2.0 Specification.

Chapter 2. Summary and Interface Specification

2.1. Summary of Changes

This specification extends the *CLARIN Federated Content Search (CLARIN-FCS) - Core 2.0* specification in the following ways:

- introducing a new query language, Chapter 4, *LexCQL*, based on Contextual Query Language (CQL) to allow querying lexical resources,
- extending the basic Hits Data View for inline markup, and adding a required Section 5.1, "Lexical Data View" for searching through lexical resources,
- extending the CLARIN-FCS *Endpoint Description* with the *Lexical Search* Capability.

2.2. Discovery

The CLARIN-FCS SRU *explain* response is extended by adding the *Lexical Search* capability and *Lexical* Data View to the *Endpoint Description* to support client auto-configuration.

2.2.1. Capabilities

The *Lexical Search* capability indicates to clients that the FCS endpoint supports searches through lexical resources using Chapter 4, *LexCQL* and serializes results in the Data Views.

Table 1. New Lexical Search Capability

Name	Capability Identifier	Summary
Lexical Search	`http://clarin.eu/fcs/capability/lex-search`	Structured search for lexical resources

2.2.2. Endpoint Description

The *Endpoint Description* will be extended by a mandatory `<ed:SupportedLexFields>` element describing supported lexical fields for querying and results. The `<ed:SupportedDataView>` element supports the additional Section 5.1, "Lexical Data View" which has its own MIME type and `@id` value.

The `<ed:EndpointDescription>` element will be extended by

- one `<ed:SupportedLexFields>` element (`REQUIRED` if Endpoint supports *Lexical Search*)

  A list of LexFields that are generally supported by this Endpoint. This list is composed of one or more `<ed:SupportedLexField>` elements. The content of a `<ed:SupportedLexField>` `MUST` be the identifier of a Lex Field (see section Section 3.2, "Field"), e.g. lemma. Each `<ed:SupportedLexField>` element `MUST` carry an `@id` attribute. The value of the `@id` attribute is later used in the `<ed:Resource>` element to indicate, which Lex Field is supported by a resource (see

below). The Lex Field identifier is used in the Lex Data View (see section Section 5.1, "Lexical Data View").

This list MUST NOT include duplicate entries, i.e. no Lex Field with the same @id identifier must appear more than once. Identifiers used in the content of the <ed:SupportedLexField> element MUST also only appear once.

The value of the @id attribute MUST NOT contain the characters , (comma) or ; (semicolon).

The <ed:Resource> element will be extended by

- one <ed:AvailableLexFields> element (REQUIRED if Endpoint supports *Lexical Search* capability)

  The <ed:AvailableLexFields> element MUST carry a @ref attribute that contains a whitespace-separated list of id values that correspond to the value of the appropriate @id attribute for the <ed:SupportedLexFields> elements that are referenced.

  In case of sub-resources, each Resource SHOULD support all Lex Fields that are supported by the parent resource. However, every resource MUST declare all available Lex Fields independently, i.e. there is no implicit inheritance semantic.

Example of an Endpoint Description with Lexical Search support

```
<ed:EndpointDescription xmlns:ed="http://clarin.eu/fcs/endpoint-description"
version="2">
  <ed:Capabilities>
    <ed:Capability>http://clarin.eu/fcs/capability/basic-search</ed:Capability>
    <ed:Capability>http://clarin.eu/fcs/capability/lex-search</ed:Capability>
  </ed:Capabilities>
  <ed:SupportedDataViews>
    <ed:SupportedDataView id="hits" delivery-policy="send-by-default">application/x-
clarin-fcs-hits+xml</ed:SupportedDataView>
    <ed:SupportedDataView id="lex" delivery-policy="send-by-default">application/x-
clarin-fcs-lex+xml</ed:SupportedDataView>
  </ed:SupportedDataViews>
  <ed:SupportedLexFields>
    <ed:SupportedLexField id="lang">lang</ed:SupportedLexField>
    <ed:SupportedLexField id="entryId">entryId</ed:SupportedLexField>
    <ed:SupportedLexField id="lemma">lemma</ed:SupportedLexField>
    <ed:SupportedLexField id="pos">pos</ed:SupportedLexField>
    <ed:SupportedLexField id="baseform">baseform</ed:SupportedLexField>
    <ed:SupportedLexField id="sentiment">sentiment</ed:SupportedLexField>
  </ed:SupportedLexFields>
  <ed:Resources>
    <ed:Resource pid="ws_sentiws">
      <ed:Title xml:lang="de">SentiWS</ed:Title>
      <ed:Title xml:lang="en">SentiWS</ed:Title>
      <ed:Description xml:lang="de">Der SentimentWortschatz, oder kurz SentiWS, ist
eine öffentlich verfügbare deutschsprachige Ressource für die Sentiment Analyse,
Opinion Mining und ähnliche Zwecke. ...</ed:Description>
      <ed:Description xml:lang="en">SentimentWortschatz, or SentiWS for short, is a
```

```
publicly available German-language resource for sentiment analysis, opinion mining
etc. ...</ed:Description>
      <ed:LandingPageURI>https://wortschatz.uni-
leipzig.de/de/download#sentiWSDownload</ed:LandingPageURI>
      <ed:Languages>
        <ed:Language>deu</ed:Language>
      </ed:Languages>
      <ed:AvailableDataViews ref="hits lex" />
      <ed:AvailableLexFields ref="lang entryId lemma pos baseform sentiment" />
    </ed:Resource>
  </ed:Resources>
</ed:EndpointDescription>
```

Example Endpoint Description for Lexical Search shows a simple Endpoint Description for an Endpoint that supports the *Lexical Search* Capability and provides the Lex Data View, which is indicated by a `<ed:SupportedDataView>` element. It only provides one top-level resource identified by the persistent identifier `ws_sentiws`. The resource has basic metadata such as multi-lingual titles, descriptions and a landing page. The predominant language in the resource contents is German. The Generic Hits Data View and the Lex Data View are supported for this resource, because the `<ed:AvailableDataViews>` element references the `<ed:SupporedDataView>` element with the `@id` with a value of `hits` and `lex`. The `<ed:AvailableLexFields>` references the Lex Fields from the list of all Lex Fields the Endpoints supports, listed in the `<ed:SupportedLexFields>` element. Each Lex Field is referenced in the space separated list of the `@ref` attribute of the `<ed:AvailableDataViews>` element pointing to the `@id` identifiers of the `<ed:SupportedLexField>` elements.

2.3. Searching and Result presentation

Queries `MUST` be formulated using the Chapter 4, *LexCQL* query language. Results `MUST` be serialized using both the *Generic Hits* Data View as defined in the CLARIN FCS Core Specification and the Section 5.1, “Lexical Data View”. For the *Generic Hits* Data View, the Section 5.2, “Extension of the Hits Data View for LexFCS” `MAY` be used, which supports limited inline markup to provide extra context to Hits.

Chapter 3. LexFCS Data Model

The *LexFCS* data model represents information about a single lexical record in a `<lex:Entry>` element and makes no general assumptions about the granularity, type or structure of the underlying lexical resource or its elements.

3.1. Entry

Each `<lex:Entry>` element consists of at least one `<lex:Field>` element, each representing a specific kind of information.

Entry attributes

The `<lex:Entry>` element `MAY` provide the default content language in IETF BCP 47 of the whole entry via the `@xml:lang` attribute. If more information is required, the `OPTIONAL` `@langUri` attribute `MAY` additionally be provided. Language specified on individual `<lex:Value>` elements overrides the `<lex:Entry>` language.

Table 2. Allowed attributes of element `<lex:Entry>`

Value Attribute	Description	Data type	Examples (without quotes)
`xml:lang`	Content language code in IETF BCP 47	String	"deu", "egy-Egyp", "cop-x-cops"
`langUri`	URI referencing information about the content language, if not expressable using BCP 47	URI	"https://en.wikipedia.org/wiki/Early_New_High_German"

3.2. Field

Each `<lex:Field>` element has a type that `MUST` be specified using a `@type` attribute. A `<lex:Field>` element of a specific type `MAY` only occur once. The `<lex:Field>` element of type `lemma` is mandatory. For an overview of all allowed field types and their restrictions, refer to Table 3, “Lexical Search Field types”.

The `@type` attribute value of a `<lex:Field>` element governs the semantics, data type, and mandatory / optional attributes of all its `<lex:Value>` elements. A `<lex:Field>` element `MUST` contain at least one `<lex:Value>` element, but `MAY` contain an unlimited number of `<lex:Value>` elements.

Table 3. Lexical Search Field types

Field Type Identifier	Short Description	Value data type	Value examples (without quotes)
`lemma`	Lemma form, **mandatory**	String	"Dog", "cat", "walking", "better"
`entryId`	Identifier of the current entry	String	"104730", "Zwahr-2049-pawk"

Field Type Identifier	Short Description	Value data type	Value examples (without quotes)
phonetic	Phonetic form	String	"t□□dā"
translation	Translation	String	"pechare" (cup)
transcription	Transcription or transliteration	String	-
Prosaic Descriptions			
definition	Definition or description	String	"A car is a vehicle with an engine […]"
etymology	Etymology information	String	-
Grammar and Morphology			
case	Morphological Case	String	"Abl", "Nom", "Tra"
number	Morphological Number	String	"Dual", "Sing", "Plur"
gender	Morphological Gender	String	"Fem", "Masc", "Com"
pos	Part-of-Speech	String	"NOUN", "NN", "N", "substantive"
baseform	Baseform (or stem …) of a lemma	String	
segmentation	Composita segmentation, hyphenation; into phonemes, syllables, subwords	String	"Schach+Brett", "to·day", "ins|ta|gram|men"
sentiment	Sentiment information	String	"angry", "positive", "-0.823"
frequency	Frequency information like occurrences, relative frequency, word rank or frequency class	String	"17234", "0.567"
Relation to other Lexical Entries			
antonym, hyponym, hypernym, meronym, holonym, synonym	Semantic relations	String	-
related	Unspecified relation	String	-
External References			
ref	A URI referencing a related resource.	URI	"http://example.org/somewhere/", "https://coptic-dictionary.org/entry.cgi?tla=C2535"
senseRef	ID of a sense definition.	String	"02961779-n", "4129315-0", "8.10"
Citations / Quotations			

Field Type Identifier	Short Description	Value data type	Value examples (without quotes)
`citation`	A citation, quotation or example of this entry's lemma.	String	"I got into my car."

3.3. Value

The `<lex:Value>` element contains the actual information content of a specific information type. Its semantics, data type, and allowed / mandatory attributes is governed by the type of the `<lex:Field>` element in which it is contained.

It is strongly encouraged to use terms of established vocabularies, where feasible. This includes in particular the use of linguistic tags and features of the Universal Dependencies annotation guidelines for the corresponding `<lex:Field>` types.

3.3.1. Generic Value attributes

A `<lex:Value>` element can be modified with additional attributes, which are, in most cases, optional but `RECOMMENDED`.

The Table 4, "Lexical Search Value attributes" lists all allowed attributes for `<lex:Value>` elements. Additional information about their usage and cases in which they are mandatory are stated below.

Table 4. Lexical Search Value attributes

Value Attribute	Description	Data type	Examples (without quotes)
`xml:id`	XML ID as target for `@idRefs` attribute	XML ID	"sense1"
`xml:lang`	Content language code in IETF BCP 47	String	"deu", "egy-Egyp", "cop-x-cops"
`langUri`	URI referencing information about the content language, if not expressable using BCP 47	URI	"https://en.wikipedia.org/wiki/Early_New_High_German"
`preferred`	`<lex:Value>` is preferred among other `<lex:Value>` element among other `<lex:Value>` elements in the same `<lex:Field>` (may be used for UI hints)	Boolean	"true"
`ref`	An unspecified reference about the content of the `<lex:Value>` element. This can be an audio file for `phonetic` field value or an external page with more information.	URI	"https://thesaurus-linguae-aegyptiae.de/sentence/ICEDBefDgQaukEMWmajL4HChr4I"
`idRefs`	Whitespace separated list of XML IDs to reference other field/values	String	"id_sense1 id_sense2"

Value Attribute	Description	Data type	Examples (without quotes)
vocabRef	Reference to the used vocabulary of the content of the <lex:Value> element	URI	"https://universaldependencies.org/u/pos"
vocabValueRef	Reference to specific value of a vocabulary	URI	"https://universaldependencies.org/u/pos/NOUN", "https://universaldependencies.org/u/feat/Gender.html#Fem"
type	A classification for a field value. It may function to group values. (UI hint)	String	"hyphenation" (segmentation), "sample" (ref)

The attributes @xml:lang and @langUri are used to specify the object language of the content of the <lex:Value> element. They override object language information provided at the <lex:Entry> element level, if any. If the @langUri attribute is given, the @xml:lang attribute is REQUIRED.

The @xml:id and @idRefs attributes are used to link <lex:Value> elements inside the same <lex:Entry> element, which can be used to highlight these relations in the user interface (like a citation <lex:Value> element referencing its corresponding definition <lex:Value> element). The semantics of this reference is unspecified.

The @type attribute SHOULD be used to further specify generic field <lex:Value> by using values from limited vocabulary to classify their intended usage. We RECOMMEND certain values for different <lex:Field> types but it is an open list. Known @type values MAY be used by Clients to interpret and process <lex:Value> contents in specific ways.

The @vocabRef attribute refers to the general vocabulary from which the content of the <lex:Field> element originates, e.g. https://universaldependencies.org/u/pos/ for referencing the POS tags of the Universal Dependencies project, which provides context for the interpretation of a string like “noun”. @vocabValueRef refers to a specific value of a vocabulary, like https://universaldependencies.org/u/pos/NOUN for a noun according to the POS tags of the Universal Dependencies project. If both are provided, they MUST refer to the same vocabulary.

3.3.2. Citation Value attributes

Values in <lex:Field> elements of type citation are allowed more optional attributes, which are listed in Table 5, “Lexical Search Value attributes for type 'citation'”

Table 5. Lexical Search Value attributes for type 'citation'

Value Attribute	Description	Data type	Examples (without quotes)
source	Name of the source from which the citation value was taken	String	"FCS Daily - Latest news and gossip stories"
sourceRef	Reference of the source from which the citation value was taken	URI	"https://wwww.fcs-daily.de/article/123"

Value Attribute	Description	Data type	Examples (without quotes)
`date`	Date information for the citation value	EDTF date/time (Level 0)	"1992-02-10", "2024", "2000-01-01T01:02:03Z"

3.3.3. Value attributes by Field type

The Table 6, "Attributes for Lexical Search Field types" lists which attributes `MUST` or `SHOULD` be attached to `<lex:Value>` elements of certain *Field Type Identifiers* to provide Clients context for interpretation of values.

Table 6. Attributes for Lexical Search Field types

Field Type Identifier	Attribute Usage Recommendation
`entryId`	The attributes `@vocabRef` or `@vocabValueRef` `MAY` be used to provide context.
`definition`, `etymology`	Hierarchical relations between `<lex:Value>` elements across `<lex:Field>` elements can be indicated by using the `@xml:id` and `@idRefs` attributes. Subordinate `<lex:Value>` elements contain an `@idRefs` attribute that refers to the `@xml:id` attribute of another `<lex:Value>` element. Clients `MAY` decide to show this hierarchy.
`pos`, `case`, `number`, `gender`	The `@vocabRef` or `@vocabValueRef` attributes `SHOULD` be used to allow disambiguation of the provided values. Clients `MAY` use them to help users, e.g. by providing translations or pointing to definitions.
`segmentation`	Values `SHOULD` use the pipe (`|`) character as separator to allow for post-processing by Clients (e.g., exchange of separators, splitting of parts). Other separator characters (`+`, `-`, `·`, …) `MAY` be used but uniform processing by Clients can not be guaranteed. The `@type` attribute `SHOULD` be used for `segmentation` `<lex:Value>` elements to indicate what type of segmentation is being performed or described.
`ref`	The `@type` attribute `MUST` be used to indicate what kind of reference is being provided.
`senseRef`	The attributes `@vocabRef` or `@vocabValueRef` `MUST` be used to provide context.
`citation`	The *special* attributes `@source`, `@sourceRef` and `@date` `SHOULD` be used to provide additional context for `citation` `<lex:Value>` elements. The `@type` attribute `SHOULD` be used to indicate what type of citation, quotation or example is being given.

3.4. Serialisation

The serialisation of a LexFCS entry is specified by the corresponding Data Views.

Chapter 4. LexCQL

LexCQL uses Contextual Query Language (CQL) to query lexical resources available in the FCS. This has the benefit of using an existing, well-known and standardized query language with an established ecosystem, including libraries, parsers and extensive documentation. The proposed Context Set can be found in Section A.1, “CQL ContextSet specification”.

LexCQL queries offer users significant compatibility and flexibility, i.e. a simple query in LexCQL SHOULD allow the retrieval of lexical records with alternative spelling or normalisation variants, e.g. upper/lower case, diacritics/umlauts, or other forms of normalization. It should, therefore, be straightforward to formulate meaningful queries, reduce frustration caused by missing or incomplete results, and also enable fuzzy search functionality. Endpoints SHOULD support this flexible, user-oriented handling, but are always free to rank more suitable results higher.

However, users SHOULD also be given the option of *"sharpening"* search queries using optional operators or modifiers, to refine queries and the associated result sets.

Full support of the LexCQL query language depends on the Endpoint. The Endpoint MUST support term-only queries. Endpoints or Clients MUST support *Level 2* of the CQL server conformance to be able to *parse* (Endpoints) or *serialize* (Clients) all of CQL and respond with appropriate error messages to the search/retrieve protocol interface. For appropriate error messages, refer to CLARIN-FCS Core 2.0, Section 3.1.

NOTE	This does not *imply* that Endpoints are *required* to support all of CQL (and LexCQL), but rather that they are able to *parse* all of CQL and generate the appropriate error message, e.g. if a query includes a feature they do not support.

LexCQL uses the same diagnostics defined in CLARIN-FCS Core 2.0 and OASIS SRU 2.0 to enable endpoints to report error and information messages.

4.1. Search term

A search term MAY be enclosed in double quotation marks ("), though it need not be. It MUST be enclosed in quotation marks if it contains any of the following characters: left or right angle bracket (<, >), left or right parenthesis ((, (), equal (=), backslash (\), apostrophe/single quotation mark ('), or whitespace. Backslash is used to escape quote and as well as itself.

Examples

- `car`
- `"car"`
- `"car wash"`
- `"car s"`
- `"27\""` to search for term `27"`
- `"\\"` to search for term `\`

4.2. Queryable Fields / Indexes

LexCQL allows querying all fields that are supported by the Chapter 3, *LexFCS Data Model* and uses their respective field types. Every LexCQL Endpoint `MUST` support queries for field `lemma` and `SHOULD` support as many queryable fields as feasible. It is recommended to use the field `lemma` as default if no field is specified in the request.

To support querying the language of a lexical record, an additional virtual field `lang` has been defined which refers to the language of the whole lexical Entry. It has the same semantics as the `@xml:lang` attribute of a Section 3.2, “Field” in a lexical Section 3.1, “Entry”, where it is queried with the `lang` relation modifier.

Examples

- `car`

  Search for lexical records with the term "car" (recommended to be interpreted as query on the field `lemma`).

- `lemma = "car"`

  Search for lexical records with the lemma "car".

- `pos = "NOUN" AND synonym = "house"`

  Search for nouns that are synonyms of "house".

- `lang = "deu" AND translation =/lang=eng "member of parliament"`

  Search for an lexical record in German that contains the English translation "member of parliament".

4.3. Relations

LexCQL supports the following relations between field and search term.

- `=`

  This is the default relation, and the server can choose any appropriate relation or means of comparing the query term with the terms from the data being searched. It is encouraged to interpret this relation in a beginner friendly way and match all reasonably similar terms, based on string normalisation, string similiarity, lemmatisation, substring etc.

- `==`

  This relation is used for exact equality matching. The term in the data is exactly equal to the term in the search query.

- `is`

  This relation can be used to search for an external value definition specified as URI.

Examples

- `lemma = "car"`

  Search for lexical records with the lemma "car". The endpoint decides which records are similar enough and might include in the result set variants, e.g. records with lemma "cars", "CARS" etc.

- `lemma == "car"`

  Search for lexical records whose lemma exactly matches the string "car".

- `pos is https://universaldependencies.org/u/pos/NOUN`

  Search for lexical records whose part-of-speech is a noun according to the POS tags of the Universal Dependencies project.

4.4. Relation Modifier

Relations `MAY` be modified with relation modifiers, each separated by a slash (`/`). The following relation modifiers are allowed. The endpoint decides if and to what degree relation modifiers of a query are considered.

- (*masked*, default modifier)

  The following patterns and special characters apply for search terms. To explicitly request this functionality, add `cql.masked` as a relation modifier.

  - A single asterisk (`*`) is used to mask zero or more characters.
  - A single question mark (`?`) is used to mask a single character, thus N consecutive question-marks means mask N characters.
  - Backslash (`\`) is used to escape `*`, `?`, quote (`"`), as well as itself. Backslash not followed immediately by one of these characters is an error.

- `unmasked`

  Do not apply masking rules, all characters are literal.

- `lang`

  Specifies the language of the requested term. It is encouraged to use and support an IETF BCP 47 compliant language code.

- `ignoreCase`, `respectCase`

  The server is instructed to either ignore or respect the case of the search term, rather than its default behavior (which is unspecified).

- `ignoreAccents`, `respectAccents`

  The server is instructed to either ignore or respect diacritics in terms, rather than its default behavior (which is unspecified, but respectAccents is recommended).

- `honorWhitespace`

  Used with `==` for exact matching to indicate that matching should even include extraneous whitespace (preceding, embedded, or following). In the absence of this modifier it is left to the server to decide whether or not to honor extraneous whitespace.

- `regexp`

  The term should be treated as a regular expression. Regular expressions are treated by the individual servers and any features beyond those found in modern POSIX regular expressions will not necessarily be supported by all servers. This modifier overrides the default 'masked' modifier, above.

- `partialMatch`, `fullMatch`

  The server is instructed that the search term is either explicitly a partial match (of a potentially longer index value) or should match the index value completely.

WARNING	The CQL Context Set defined in OASIS specifies the following modifiers that will not be used for LexCQL: `word`, `string` (how term should be tokenized for matching, `string` for *no break into words*)

Examples

- `lemma = "car s*"`

  Search for lexical records whose lemma contains the string "car s" followed by any number of characters (like "car s", "car service" or "car sickness").

- `lemma =/unmasked "car s*"`

  Search for lexical records whose lemma is "car s*".

- `synonym =/lang=eng/ignoreCase "handy"`

  Search for lexical records with the synonym "handy" (including all case variations, like "HaNdY") in English language.

4.5. Operators

LexCQL supports the following Boolean operators to form complex queries. Boolean operators have the same precedence; they are evaluated left-to-right. Parentheses may be used to override left-to-right evaluation.

- `AND`

  The set of records representing two search clauses linked by `AND` is the intersection of the two sets of records representing the two search clauses.

- `OR`

  The set of records representing two search clauses linked by `OR` is the union of the two sets of records representing the two search clauses.

- `NOT`

  The set of records representing two search clauses linked by `NOT` is the set of records representing the left hand set which are not in the set of records representing the right hand set. `NOT` cannot be used as a unary operator, it is interpreted as "AND NOT".

Examples

- `lemma = "car" AND pos = "NOUN"`

  Search for lexical records with both the lemma "car" and the part-of-speech "NOUN".

- `lemma = "car" AND (pos = "NOUN" OR pos = "ADJ")`

  Search for lexical records with the lemma "car" and either the part-of-speech "NOUN" or "ADJ".

- `lemma = car NOT pos = "NOUN"`

  Search for lexical records with the lemma "car" while not having the part-of-speech "NOUN".

- `pos = NOUN OR verb`

  Search for lexical records with the search clauses `pos = NOUN` and `verb`, connected with a Boolean OR operator.

Chapter 5. LexFCS Data Views

Data formats for the representation of results.

5.1. Lexical Data View

The *Lexical (LEX)* Data View is the mandatory serialization of search results for *Lexical Search* queries. It structures information into key and multiple values pairs. More details in Data Model.

Description	The representation of a lexical resource
MIME type	`application/x-clarin-fcs-lex+xml`
Payload Disposition	*inline*
Payload Delivery	*send-by-default* (`REQUIRED`)
Recommended Short Identifier	`lex` (`RECOMMENDED`)
XML Schema	DataView-Lex.xsd

The *Lexical (LEX)* Data View is serialized as XML in the `<fcs:DataView>` element, specified in FCS Core 2.0 specification. (section "Result Format", §2.2.3). The elements `<lex:Entry>`, `<lex:Field>` and `<lex:Value>` with their attributes are the direct serialization of the Data Model.

A minimal example can be seen at Example of basic **Lex** Data View. A few more targeted examples can be found in Section B.1.1, “Serialization with the Lexical Data View” demonstrating edge cases and special features when using this Data View.

Example of basic **Lex** *Data View*

```
<!-- potential @pid and @ref attributes omitted -->
<fcs:DataView type="application/x-clarin-fcs-lex+xml"
xmlns:fcs="http://clarin.eu/fcs/resource">
  <lex:Entry xmlns:lex="http://clarin.eu/fcs/dataview/lex">
    <lex:Field type="lemma">
      <lex:Value xml:lang="deu">Becher</lex:Value>
    </lex:Field>
    <!-- ⋯ -->
  </lex:Entry>
</fcs:DataView>
```

Serialization examples in the *Lexical (LEX)* Data View can be found at: https://gitlab.gwdg.de/textplus/ag-fcs-lex-fcs-dataview/-/tree/master/examples.

5.2. Extension of the Hits Data View for LexFCS

The *Generic Hits (HITS)* Data View is mandatory in FCS Core 2.0 specification (section "Basic Search", §2.2.3.2). This schema extends the `<hits:Hit>` element with an optional `@kind` attribute (see

Section 5.2.1, “Using the @kind Attribute”), which provides information on the content of the Hit result.

Description	The representation of the hit
MIME type	`application/x-clarin-fcs-hits+xml`
Payload Disposition	*inline*
Payload Delivery	*send-by-default* (`REQUIRED`)
Recommended Short Identifier	`hits` (`RECOMMENDED`)
XML Schema	DataView-LexHits.xsd, based on "DataView-Hits.xsd"

Example of basic **Hits** *Data View*

```
<!-- potential @pid and @ref attributes omitted -->
<fcs:DataView type="application/x-clarin-fcs-hits+xml"
xmlns:fcs="http://clarin.eu/fcs/resource">
  <hits:Result xmlns:hits="http://clarin.eu/fcs/dataview/hits">The quick brown
<hits:Hit>fox</hits:Hit> jumps over the lazy<hits:Hit>dog</hits:Hit>.</hits:Result>
</fcs:DataView>
```

5.2.1. Using the @kind Attribute

To extend the Hits Data View, the `<hits:Hit>` element is reused as per the FCS Core 2.0 Specification, but with the additional optional `@kind` attribute, which provides content hinting. The values of the `@kind` attribute `@kind` attribute follow the scheme `lex-<FIELD-TYPE>`, e.g. `lex-lemma` for lemma, `lex-pos` for part of speech (POS) tags etc. Textual content outside of `<hits:Hit>` are displayed unchanged.

Example of extended **Hits** *Data View with additional* `@kind` *attributes*

```
<fcs:DataView type="application/x-textplus-fcs-hits+xml"
xmlns:fcs="http://clarin.eu/fcs/resource">
  <hits:Result xmlns:hits="http://textplus.org/fcs/dataview/hits"><hits:Hit kind="lex-
lemma">Apple</hits:Hit>: <hits:Hit kind="lex-pos">NOUN</hits:Hit>. <hits:Hit
kind="lex-def">An apple is an edible fruit produced by an apple
tree.</hits:Hit></hits:Result>
</fcs:DataView>
```

Endpoints `MUST` generate responses that are valid according to the XML schema "DataView-LexHits.xsd".

A. Normative Appendix

A.1. CQL ContextSet specification

Used identifier: `http://text-plus.org/cql/lexres/1.0/` (*draft*)

Recommended prefix: `lexres`

> **TIP** For more examples of CQL ContextSets, see the list of Context Sets at the Library of Congress (LoC).

A.1.1. Indexes

> **TIP** For more information about CQL indexes, see The CQL Context Set, section "INDEXES".

Index title	Description
`lemma`	Lemma form
`entryId`	Identifier of the current entry
`phonetic`	Phonetic form
`translation`	Translation
`transcription`	Transcription or transliteration
`definition`	Definition or description
`etymology`	Etymology information
`case, number, gender`	Morphological information like case, number, gender, e.g. in full-form dictionaries
`pos`	Part of Speech
`baseform`	Baseform (stem, root, …) of a lemma
`segmentation`	Composita segmentation, hyphenation; into phonemes, syllables, subwords
`sentiment`	Sentiment information
`frequency`	Frequency information like occurrences, relative frequency, word rank or frequency class
`synonym, antonym, hyponym, hypernym, meronym, holonym`	Semantic relations, analogous to types in TEI Lex0
`related`	Unspecified (semantic) relation
`ref`	A URI referencing a related resource.
`senseRef`	ID of a sense Sense, Entity, … definition
`citation`	A citation, quotation or example of this entry's lemma.

A.1.2. Relations

> **TIP** More information about CQL relations can be found The CQL Context Set, section "RELATIONS".

A.1.2.1. Implicit Relations

- `=`

  alias: scr

  Different functions based on index. Suggested use:

  - *Full match* as default behaviour,
  - *Full text search* for longer text fields like `definition`, `etymology` and `citation`.

  Endpoints `MAY` decide whether they prefer to focus on precision or recall when matching, e.g. returning case-insensitive matches or other normalizations.

- `==`

  alias: exact

  Exact equality matching. The term in the data is exactly equal to the term in the search query.

A.1.2.2. Defined Relations

- `is`

  To query an identifier or entity instead of the string value (`=`).

  - `pos is "https://universaldependencies.org/u/pos/NOUN"` (searching for the `https://universaldependencies.org/u/pos/NOUN` concept)
    vs. `pos = "NOUN"` (searching for the `NOUN` string value)

A.1.3. Booleans

> **TIP** For more information about Booleans in CQL, see The CQL Context Set, section "BOOLEANS".

- `AND`
- `OR`
- `NOT`
- `PROX` (*is not used!*)

A.1.4. Relation Modifiers / Relation Qualifiers

> **TIP** For more information about relation modifiers/qualifiers, see The CQL Context Set,

> section "RELATION MODIFIERS".

A.1.4.1. Functional Modifiers

- `lang=value`

  Specification of a ISO639/BCP47 language tag of the search term. This can for example be used to search for translations in certain languages.

- `ignoreCase`, `respectCase`

  The server is instructed to either ignore or respect the case of the search term, rather than its default behavior (which is unspecified).

- `ignoreAccents`, `respectAccents`

  The server is instructed to either ignore or respect diacritics in terms, rather than its default behavior (which is unspecified, but respectAccents is recommended).

- `partialMatch`, `fullMatch`

  The server is instructed that the search term is either explicitly a partial match (of a potentially longer index value) or should match the index value completely.

A.1.4.2. Matching

- (`masked`, default modifier)

  The following patterns and special characters apply for search terms. To explicitly request this functionality, add `cql.masked` as a relation modifier.

  - A single asterisk (`*`) is used to mask zero or more characters.
  - A single question mark (`?`) is used to mask a single character, thus N consecutive question-marks means mask N characters.
  - Backslash (`\`) is used to escape `*`, `?`, quote (`"`), as well as itself. Backslash not followed immediately by one of these characters is an error.

- `unmasked`

  Do not apply masking rules, all characters are literal.

- `honorWhitespace`

  Used with `==` for exact matching to indicate that matching should even include extraneous whitespace (preceding, embedded, or following). In the absence of this modifier it is left to the server to decide whether or not to honor extraneous whitespace.

- `regexp`

  The term should be treated as a regular expression. Regular expressions are treated by the individual servers and any features beyond those found in modern POSIX regular expressions

will not necessarily be supported by all servers. This modifier overrides the default 'masked' modifier, above.

A.1.5. Boolean Modifiers

> **TIP** More information about Boolean modifiers can be found The CQL Context Set, section "BOOLEAN MODIFIERS".

none

A.1.6. Examples

1. `cat`
   `"cat"`
   `"United Nations"`
   Different variants to search in the default index.
2. `lemma == mouse`
   Search for exact string value `mouse` in `lemma` index.
3. `pos = ADJ`
   Search for adjectives.
4. `definition = "cat"`
   Search for records whose definition contains the term "cat".
5. `pos = "NOUN" NOT "lion" AND definition = carnivore`
   Search for nouns with "carnivore" in definition; exclusion of records with "lion".
6. `pos = NOUN AND ( lemma = Apfel OR lemma = "Birne" )`
   Search for nouns that have lemma with either "Apfel" or "Birne".
7. `translation =/lang=eng car`
   Search for lexical entries with an English translation "car".
8. `lemma =/unmasked "^ca?r*"`
   Search for the lemma with the literal term "^ca?r*".

B. Non-Normative Appendix

B.1. Best Practices

B.1.1. Serialization with the Lexical Data View

B.1.1.1. Specifying a default language — @xml:lang and @langUri

Each `<lex:Value>` element can specify language information using the `@xml:lang` and `@langUri` attributes. To avoid redundancy, a default language MAY be specified at the `<lex:Entry>` element. If `<lex:Value>` elements do not specify their language explicitely, they inherit the entry`s language information.

`<lex:Value>` elements with the same `@xml:lang` attribute value but different `@langUri` attribute values MUST be considered to describe different languages. As a consequence, only `<lex:Value>` elements with the same `@xml:lang` and `@langUri` attribute values as the `<lex:Entry>` element MAY inherit the entry`s language information. If in doubt, specify the language attributes redundantly.

Default and explicit language specification for Values

```
<!-- abbreviated example from examples/dwee-Becher.lex.xml -->
<lex:Entry xmlns:lex="http://clarin.eu/fcs/dataview/lex" xml:lang="deu"> ①
  <lex:Field type="lemma">
    <lex:Value xml:lang="deu">Becher</lex:Value>
  </lex:Field>
  <lex:Field type="segmentation">
    <lex:Value type="hyphenation">Be|cher</lex:Value> ②
  </lex:Field>
  <lex:Field type="hypernym">
    <lex:Value xml:lang="deu">Gefäß</lex:Value> ③
    <lex:Value>Gegenstand</lex:Value>
    <lex:Value xml:lang="goh">pehhari</lex:Value>
  </lex:Field>
  <lex:Field type="synonym">
    <lex:Value>Eimer</lex:Value>
    <lex:Value xml:lang="deu"
  langUri="https://en.wikipedia.org/wiki/Early_New_High_German">Kraus</lex:Value>  ④
    <lex:Value xml:lang="goh">kelich</lex:Value>
  </lex:Field>
</lex:Entry>
```

① Specifying German as default language of `<lex:Entry>` via `@xml:lang`.

② Any `<lex:Value>` element without its own language attributes inherits language information from its parent `<lex:Entry>`, here `@xml:lang="deu"`.

③ `@xml:lang="deu"` is redundant.

④ Due to `@langUri`, the content language of this `<lex:Value>` element MUST be considered to be a different language, even though the `@xml:lang` attribute value is the same as the one at

`<lex:Entry>` level. As no language information is inherited here, both attributes `@xml:lang` and `@langUri` `MUST` be specified.

B.1.1.2. Contextualize Value contents for improved interpretability

The example Contextualization of senseRef Values shows how plain text content can be semantically enriched by explicitly stating the used vocabulary or by referencing the value's external definition via attributes `@vocabRef` or `@vocabValueRef`.

Contextualization of senseRef Values

```
<!-- abbreviated example from examples/wortschatz-Auto.lex.xml -->
<lex:Entry xmlns:lex="http://clarin.eu/fcs/dataview/lex" xml:lang="deu">
  <lex:Field type="lemma">
    <lex:Value xml:lang="deu">Auto</lex:Value>
  </lex:Field>
  <lex:Field type="senseRef">
    <!-- Dornseiff -->
    <lex:Value vocabRef="https://doi.org/10.1515/9783110457742">8.10</lex:Value> ①
    <!-- GermaNet -->
    <lex:Value vocabRef="http://textplus.sfs.uni-tuebingen.de/api/germanet/synset"
>s123456789</lex:Value>
    <!-- Gemeinsame Normdatei (GND) -->
    <lex:Value vocabValueRef="http://d-nb.info/gnd/4129315-0">4129315-0</lex:Value>
 ②
    <!-- Princeton WordNet -->
    <lex:Value vocabRef="http://wordnet-rdf.princeton.edu/ontology#Synset"
>02961779-n</lex:Value> ③
  </lex:Field>
</lex:Entry>
```

① Using the `@vocabRef` attribute referencing `https://doi.org/10.1515/9783110457742`, the value `8.10` can be interpreted as category *"8.10 Auto, Fahrt"* of the Dornseiff dictionary, volume 9.

② The `@vocabValueRef` attribute value contains the authoritative reference at the authority file GND for the value `4129315-0` .

③ Using the `@vocabRef` attribute, the value `02961779-n` can be interpreted in the context of Princeton WordNet synsets.

B.1.1.3. Connecting Values using @xml:id and @idRefs

Any `<lex:Value>` element can refer to other `<lex:Value>` elements in the same `<lex:Entry>` element by specifying their IDs – stated in their `@xml:id` attribute – in a whitespace separated list in the `@idRefs` attribute. The type of this relation is unspecified.

Value relations using @xml:id and @idRefs

```
<!-- abbreviated example from examples/GermaNet_Ei.lex.xml -->
<lex:Entry xmlns:lex="http://clarin.eu/fcs/dataview/lex" xml:lang="deu">
  <lex:Field type="lemma">
    <lex:Value xml:lang="deu">Ei</lex:Value>
```

```
  </lex:Field>
  <lex:Field type="senseRef">
    <lex:Value xml:id="sense_1" vocabRef="http://textplus.sfs.uni-
tuebingen.de/api/germanet/synset">s39427</lex:Value> ①
    <lex:Value xml:id="sense_2" vocabRef="http://textplus.sfs.uni-
tuebingen.de/api/germanet/synset">s25806</lex:Value>
    <lex:Value xml:id="sense_3" vocabRef="http://textplus.sfs.uni-
tuebingen.de/api/germanet/synset">s25813</lex:Value>
  </lex:Field>
  <lex:Field type="definition">
    <lex:Value idRefs="sense_1">unzählbar, ohne Plural: […]</lex:Value> ②
    <lex:Value idRefs="sense_2">ein Schalengebilde, in dem […]</lex:Value>
    <lex:Value idRefs="sense_3">eine Keimzelle</lex:Value>
  </lex:Field>
  <lex:Field type="hypernym">
    <lex:Value idRefs="sense_1">festes Nahrungsmittel</lex:Value> ②
    <lex:Value idRefs="sense_2">Keim</lex:Value>
    <lex:Value idRefs="sense_2">Keimling</lex:Value>
    <lex:Value idRefs="sense_3">Gamet</lex:Value>
  </lex:Field>
  <lex:Field type="hyponym">
    <lex:Value idRefs="sense_2">Brutei</lex:Value>
    <lex:Value idRefs="sense_2 sense_3">Windei</lex:Value> ③
  </lex:Field>
</lex:Entry>
```

① `<lex:Value>` element with XML ID `sense_1`.

② `<lex:Value>` elements referring to the `<lex:Value>` element with `@xml:id="sense_1"` attribute.

③ A `<lex:Value>` elements refering to multiple XML IDs.

B.1.1.4. Connecting Values within Fields using @xml:id and @idRefs to build Value hierarchies

`<lex:Value>` elements within `<lex:Field>` elements of type *definition*, *etymology* and *senseRef* can be organized hierarchically by using the `@xml:id` and `@idRefs` attributes. `<lex:Value>` elements can refer to their head `<lex:Value>` element via their `@idRefs` attribute.

If `@idRefs` contains multiple IDs then only the first one is assumed to be the direct parent `<lex:Value>` element, so additional IDs may still connect the `<lex:Value>` element to other `<lex:Value>` elements as described in Section B.1.1.3, “Connecting Values using @xml:id and @idRefs”.

Value hierarchy using @xml:id and @idRefs

```
<!-- abbreviated example from examples/dwdswb-herausgehen.lex.xml -->
<lex:Entry xmlns="http://clarin.eu/fcs/dataview/lex" xml:lang="deu">
  <lex:Field type="lemma">
    <lex:Value xml:lang="deu">herausgehen</lex:Value>
  </lex:Field>
  <lex:Field type="definition">
    <lex:Value xml:id="def-1">(zu Fuß) (von irgendwo, drinnen) heraus bzw. nach
```

```
draußen, […]</lex:Value> ①
    <lex:Value xml:id="def-1-1" idRefs="def-1">etw. (eine Situation, einen Zustand) hinter […]</lex:Value> ②
    <lex:Value xml:id="def-2">sich (aus einem Objekt) herauslösen oder entfernen lassen</lex:Value>
    <lex:Value xml:id="def-3">sich ableiten, herausfiltern, schlussfolgern lassen</lex:Value>
    <lex:Value xml:id="def-5">besonders von postalischen oder elektronischen Sendungen; […]</lex:Value>
    <lex:Value xml:id="def-5-1" idRefs="def-5">von Informationen; gestreut , verbreitet […]</lex:Value>
  </lex:Field>
  <lex:Field type="citation">
    <lex:Value idRefs="def-1">ich sah sie aus dem Garten herausgehen</lex:Value>
    <lex:Value idRefs="def-1" source="Die Welt, 29.02.2020">Nach etwa 350 Metern […]</lex:Value> ③
    <lex:Value idRefs="def-1-1" source="Neue Westfälische, 08.05.2023">Die Kliniken […]</lex:Value>
  </lex:Field>
</lex:Entry>
```

① A `<lex:Value>` element with an `@xml:id` attribute of value `def-1`.

② A `<lex:Value>` element stating in its `@idRefs` attribute to be a child of the `<lex:Value>` element with `@xml:id` attribute of value `def-1`.

③ `<lex:Value>` elements in `<lex:Field>` elements of type other than `definition` or `etymology` can refer to the same `<lex:Value>` element but will not be seen as children of this hierarchy, only as related in an unspecifed way.

B.1.1.5. Referencing external resources using the @ref attribute

The `@ref` attribute is used to reference an external resource that provides additional information about the `<lex:Value` element's content. Depending on the `<lex:Field>` type, the referenced resource may be interpreted differently.

For example in `<lex:Value>` elements in `<lex:Field>` element with `@type="phonetic"`, the external resource might provide a link to an audio file with pronunciation.

Audio integration for phonetic Values

```
<!-- abbreviated example from examples/dwdswb-herausgehen.lex.xml -->
<lex:Entry xmlns="http://clarin.eu/fcs/dataview/lex" xml:lang="deu">
  <lex:Field type="lemma">
    <lex:Value xml:lang="deu">herausgehen</lex:Value>
  </lex:Field>
  <lex:Field type="phonetic">
    <lex:Value ref="https://www.dwds.de/audio/032/herausgehen.mp3">
h□□□a□□sge□□n</lex:Value>
  </lex:Field>
</lex:Entry>
```

Changelog

2025-06-20 — Publication of LexFCS (v0.3)

- Specify *Lex Fields* in the *Endpoint Description* for client self-configuration, describing what lex fields an endpoint supports for querying and that can appear in results (Lex Data View)
- Change Lex Data View namespace to `http://clarin.eu/fcs/dataview/lex`
- Update Lex Field Types
- Various fixes

2024-12-16 — Publication of LexFCS (v0.2)

- Specify *Lex Data Model* with *Lex Data View* serialization
- LexCQL for new *Lex Data Model*

2024-04-10 — Publication on CLARIN-ERIC GitHub

2023-05-09 — Publication of first draft (v0.1) on Zenodo

- Propose *LexCQL* query language and extension of *Hits Data View* for LexFCS
- LexFCS CQL Context Set for *LexCQL*