# Federated Content Search for Lexical Resources (LexFCS)

## *Specification*

Erik Körner, Thomas Eckart, Axel Herold, Frank Wiegand, Frank Michaelis,
Matthias Bremm, Louis Cotgrove, Thorsten Trippel, Felix Rau

Version 0.1, 2023-05-04

# 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.

**Aggregator**
A module or service to dispatch queries to repositories and collect results.

**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 information.

**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**
A piece of data returned by a Search Engine that matches the search criterion. What is considered a Hit highly depends on Search Engine.

**Interface Specification**
Common harmonized interface and suite of protocols that repositories need to implement.

**PID**

A Persistent identifier is a long-lasting reference to a digital object.

**Repository Registry**

A separate service that allows registering Repositories and their Endpoints and provides information about these to other components, e.g. an Aggregator. The CLARIN Center Registry is an implementation of such a repository registry.

**Resource**

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

**Resource Fragment**

A smaller unit in a Resource, e.g. a sentence in a text corpus or a time interval in an audio transcription.

**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

## 1.4. Non-Normative References

**UD-POS**

Universal Dependencies, Universal POS tags v2.0, https://universaldependencies.org/u/pos/

## 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 `MUST 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 element content or attribute value.

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.

[1] SHOULD be used as default query field

# Chapter 2. LexCQL

The LexCQL makes use of the Contextual Query Language (CQL) for queries on lexical resources in the context of 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 CQL ContextSet specification.

LexCQL queries should offer the greatest possible compatibility and flexibility to users, i.e. it `SHOULD` allow the retrieval of lexical records which are available in different spelling or normalisation variants (upper/lower case; diacritics/umlauts; other forms of normalization) with a simple query. It therefore should make it easier 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.

## 2.1. Queryable fields / indexes

Every LexCQL Endpoint `MUST` support queries for field `lemma` (at least implicitly as default field) and `SHOULD` support as many queryable fields as feasible.

### 2.1.1. Form and sense definition

- `lemma`: Lemma, main form, article name [1]
- `def`: Definition

### 2.1.2. Grammatical properties (WIP)

NOTE Analogous to TEI Lex-0 types .

- `pos`: Part of speech [1 - Introduction]

NOTE *Open question*: all TEI Lex-0 vocabulary (like case, gender, etc.)?

### 2.1.3. Semantic relations / Cross-references *(WIP)*

NOTE Analogous to TEI Lex-0 types with `xr` Context Set.

- `xr$synonymy`: Synonym
- `xr$hyponymy`: Hyponym
- `xr$hypernymy`: Hyperonym
- `xr$meronymy`: Meronym
- `xr$antonymy`: Antonym

**NOTE** *Open question*: how exactly do search scenarios look like here?

**Examples**

- `pos = "NOUN" AND xr$synonymy = "house"`

  Searching for nouns that are synonyms to "house".

### 2.1.4. Senses, Entities, Synsets *(WIP)*

- `senseRef`: References to external sense definitions (like Princeton WordNet, GermaNet, Interlingual Index, authority files)

**Examples**

- `senseRef = "https://d-nb.info/gnd/118571249"`

  Searching for all references of the entity https://d-nb.info/gnd/118571249 (person "Gottfried Wilhelm Leibniz").

- `senseRef = "wordnet:study%2:31:02::"`

  Searching for sense "study" (`study%2:31:02::`*, analyze, analyse, study, examine, canvass*).

  **NOTE** Some inventory of authorities or lookups should be defined and agreed on. Otherwise, it is up to the Endpoint to interpret the senses correctly.

*Open questions*:

- Standard way of referencing Princeton WordNet senses?

### 2.1.5. Operators *(WIP)*

- Partial or full match of an *index*
  - `=` for full match?

    **NOTE** Very generic definition in CQL Context Set specification.

    *Open questions*:

    - Taking into account lower/uppercase normalisation of Umlauts?
    - CQL operators for partial matches or usage of `/approx`?
    - Regular expressions?
- Combination with `AND` / `OR` and parentheses `(` … `)`

  **NOTE** Leave full support to the endpoints!

[0] Potential use of the Universal Dependencies POS tags

# Chapter 3. LexFCS Data Views

Data formats for result representation.

## 3.1. Extension of the Hits Data View for LexFCS

Based on:

- FCS Core 2.0 specification (section "Basic Search", §2.2.3.2)
- Hits XML schema "DataView-Hits.xsd"

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

```
<!-- potential @pid and @ref attributes omitted -->
<fcs:DataView type="application/x-clarin-fcs-hits+xml">
  <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>
```

Reuse of the `<hits:Hit>` element, with the extension of content hinting by using an optional attribute `@kind` with the following allowed values:

- `lex-lemma`: Lemma,
- `lex-pos`: Part of speech,
- `lex-def`: Definition.

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">
  <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 | Details/Open questions |
|---|---|---|
| `lemma` | Lemma name | Support of multiword expressions? |
| `pos` | Part of Speech | Use of Universal POS tags. Allow custom tag sets? |
| `def` | Definition | |
| `xr$synonymy`, … | Semantic relations, analogous to types in TEI Lex0 | Only *Synonym*, *Hyponym*, *Hyperonym*, *Meronym*, *Antonym*, or additional relations? |
| `senseRef` | Sense, Entity, … - URI/ID pointint to unique identifier, e.g. for disambiguation | Any URI allowed? Defined subsets (using prefixes)? Subdivision by type (e.g. synset/sense or entity, …)? |

### 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

- `=`

  Different functions based on index. Suggested use:

  - *Full match* for `lemma` and `def`,
  - for other indexes (like `pos`) use as *"contained in"* (*Endpoint-dependent behaviour*).

#### A.1.2.2. Defined Relations

(potential definition of full/partial match in the future)

### A.1.3. Booleans

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

- `AND`
- `OR`
- `NOT` (*might be unnecessary*)
- `PROX` (*might be obsolete*)

### A.1.4. Relation Modifiers / Relation Qualifiers

TIP For more information about relation modifiers/qualifiers, see The CQL Context Set, section "RELATION MODIFIERS".

**none**

- `=` with `/contains`, `/startswith`, `/endswith`, `/fullmatch`, `/partialmatch` modifiers

### 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"`
   Searching in the default index; `cql.word` or `lemma`
2. `pos = ADJ`
   Search for adjectives
3. `def = "cat"`
   Search for records whose definition contains the term "cat".
   TODO: Stemming, Lower/Uppercase, Subword-Matches
4. `pos = NOUN NOT lion AND def = carnivore`
   Search for nouns with "carnivore" in definition; exclusion of records with "lion"