searchRetrieve: Part 5. CQL: The Contextual Query Language Version 1.0

Committee Specification 01

13 April 2012

Specification URIs

This version:

http://docs.oasis-open.org/search-ws/searchRetrieve/v1.0/cs01/part5-cql/searchRetrieve-v1.0-cs01-part5-cql.doc (Authoritative)
http://docs.oasis-open.org/search-ws/searchRetrieve/v1.0/cs01/part5-cql/searchRetrieve-v1.0-cs01-part5-cql.html
http://docs.oasis-open.org/search-ws/searchRetrieve/v1.0/cs01/part5-cql/searchRetrieve-v1.0-cs01-part5-cql.pdf

Previous version:

N/A

Latest version:

http://docs.oasis-open.org/search-ws/searchRetrieve/v1.0/searchRetrieve-v1.0-part5-cql.doc (Authoritative)
http://docs.oasis-open.org/search-ws/searchRetrieve/v1.0/searchRetrieve-v1.0-part5-cql.html
http://docs.oasis-open.org/search-ws/searchRetrieve/v1.0/searchRetrieve-v1.0-part5-cql.pdf

Technical Committee:

OASIS Search Web Services TC

Chairs:

Ray Denenberg (rden@loc.gov), Library of Congress
Matthew Dovey (m.dovey@jisc.ac.uk), JISC Executive, University of Bristol

Editors:

Ray Denenberg (rden@loc.gov), Library of Congress
Larry Dixson (ldix@loc.gov), Library of Congress
Ralph Levan (levan@oclc.org), OCLC
Janifer Gatenby (Janifer.Gatenby@oclc.org), OCLC
Tony Hammond (t.hammond@nature.com), Nature Publishing Group
Matthew Dovey (m.dovey@jisc.ac.uk), JISC Executive, University of Bristol

Additional artifacts:

This prose specification is one component of a Work Product which also includes:

- XML schemas: http://docs.oasis-open.org/search-ws/searchRetrieve/v1.0/cs01/schemas/
- *searchRetrieve: Part 0. Overview Version 1.0*. http://docs.oasis-open.org/search-ws/searchRetrieve/v1.0/cs01/part0-overview/searchRetrieve-v1.0-cs01-part0-overview.html
- *searchRetrieve: Part 1. Abstract Protocol Definition Version 1.0*. http://docs.oasis-open.org/search-ws/searchRetrieve/v1.0/cs01/part1-apd/searchRetrieve-v1.0-cs01-part1-apd.html
- *searchRetrieve: Part 2. searchRetrieve Operation: APD Binding for SRU 1.2 Version 1.0*. http://docs.oasis-open.org/search-ws/searchRetrieve/v1.0/cs01/part2-sru1.2/searchRetrieve-v1.0-cs01-part2-sru1.2.html

- *searchRetrieve: Part 3. searchRetrieve Operation: APD Binding for SRU 2.0 Version 1.0*. http://docs.oasis-open.org/search-ws/searchRetrieve/v1.0/cs01/part3-sru2.0/searchRetrieve-v1.0-cs01-part3-sru2.0.html
- *searchRetrieve: Part 4. APD Binding for OpenSearch Version 1.0*. http://docs.oasis-open.org/search-ws/searchRetrieve/v1.0/cs01/part4-opensearch/searchRetrieve-v1.0-cs01-part4-opensearch.html
- *searchRetrieve: Part 5. CQL: The Contextual Query Language Version 1.0*. (this document) http://docs.oasis-open.org/search-ws/searchRetrieve/v1.0/cs01/part5-cql/searchRetrieve-v1.0-cs01-part5-cql.html
- *searchRetrieve: Part 6. SRU Scan Operation Version 1.0*. http://docs.oasis-open.org/search-ws/searchRetrieve/v1.0/cs01/part6-scan/searchRetrieve-v1.0-cs01-part6-scan.html
- *searchRetrieve: Part 7. SRU Explain Operation Version 1.0*. http://docs.oasis-open.org/search-ws/searchRetrieve/v1.0/cs01/part7-explain/searchRetrieve-v1.0-cs01-part7-explain.html

Related work:

This specification is related to:

- CQL: Contextual Query Language. Library of Congress. http://www.loc.gov/standards/sru/specs/cql.html

Abstract:

This is one of a set of documents for the OASIS Search Web Services (SWS) initiative. CQL, the *Contextual Query Language*, is a formal language for representing queries to information retrieval systems. Its objective is to combine simplicity with expressiveness, to accommodate the range of complexity from very simple queries to very complex. CQL queries are intended to be human readable and writable, intuitive, and expressive.

Status:

This document was last revised or approved by the OASIS Search Web Services TC on the above date. The level of approval is also listed above. Check the "Latest version" location noted above for possible later revisions of this document.

Technical Committee members should send comments on this specification to the Technical Committee's email list. Others should send comments to the Technical Committee by using the "Send A Comment" button on the Technical Committee's web page at http://www.oasis-open.org/committees/search-ws/.

For information on whether any patents have been disclosed that may be essential to implementing this specification, and any offers of patent licensing terms, please refer to the Intellectual Property Rights section of the Technical Committee web page (http://www.oasis-open.org/committees/search-ws/ipr.php).

Citation format:

When referencing this specification the following citation format should be used:

[SearchRetrievePt5]

searchRetrieve: Part 5. CQL: The Contextual Query Language Version 1.0. 13 April 2012. OASIS Committee Specification 01. http://docs.oasis-open.org/search-ws/searchRetrieve/v1.0/cs01/part5-cql/searchRetrieve-v1.0-cs01-part5-cql.html.

Notices

All capitalized terms in the following text have the meanings assigned to them in the OASIS Intellectual Property Rights Policy (the "OASIS IPR Policy"). The full Policy may be found at the OASIS website.

This document and translations of it may be copied and furnished to others, and derivative works that comment on or otherwise explain it or assist in its implementation may be prepared, copied, published, and distributed, in whole or in part, without restriction of any kind, provided that the above copyright notice and this section are included on all such copies and derivative works. However, this document itself may not be modified in any way, including by removing the copyright notice or references to OASIS, except as needed for the purpose of developing any document or deliverable produced by an OASIS Technical Committee (in which case the rules applicable to copyrights, as set forth in the OASIS IPR Policy, must be followed) or as required to translate it into languages other than English.

The limited permissions granted above are perpetual and will not be revoked by OASIS or its successors or assigns.

This document and the information contained herein is provided on an "AS IS" basis and OASIS DISCLAIMS ALL WARRANTIES, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO ANY WARRANTY THAT THE USE OF THE INFORMATION HEREIN WILL NOT INFRINGE ANY OWNERSHIP RIGHTS OR ANY IMPLIED WARRANTIES OF MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE.

OASIS requests that any OASIS Party or any other party that believes it has patent claims that would necessarily be infringed by implementations of this OASIS Committee Specification or OASIS Standard, to notify OASIS TC Administrator and provide an indication of its willingness to grant patent licenses to such patent claims in a manner consistent with the IPR Mode of the OASIS Technical Committee that produced this specification.

OASIS invites any party to contact the OASIS TC Administrator if it is aware of a claim of ownership of any patent claims that would necessarily be infringed by implementations of this specification by a patent holder that is not willing to provide a license to such patent claims in a manner consistent with the IPR Mode of the OASIS Technical Committee that produced this specification. OASIS may include such claims on its website, but disclaims any obligation to do so.

OASIS takes no position regarding the validity or scope of any intellectual property or other rights that might be claimed to pertain to the implementation or use of the technology described in this document or the extent to which any license under such rights might or might not be available; neither does it represent that it has made any effort to identify any such rights. Information on OASIS' procedures with respect to rights in any document or deliverable produced by an OASIS Technical Committee can be found on the OASIS website. Copies of claims of rights made available for publication and any assurances of licenses to be made available, or the result of an attempt made to obtain a general license or permission for the use of such proprietary rights by implementers or users of this OASIS Committee Specification or OASIS Standard, can be obtained from the OASIS TC Administrator. OASIS makes no representation that any information or list of intellectual property rights will at any time be complete, or that any claims in such list are, in fact, Essential Claims.

The name "OASIS" is a trademark of OASIS, the owner and developer of this specification, and should be used only to refer to the organization and its official outputs. OASIS welcomes reference to, and implementation and use of, specifications, while reserving the right to enforce its marks against misleading uses. Please see http://www.oasis-open.org/who/trademark.php for above guidance.

Table of Contents

1 Introduction

This is one of a set of documents for the OASIS Search Web Services (SWS) initiative.

This document is “CQL: The Contextual Query Language”.

The documents in this collection of specifications are:

1. Overview
2. APD
3. SRU1.2
4. SRU2.0
5. OpenSearch
6. CQL (this document)
7. Scan
8. Explain

The Abstract Protocol Definition (APD) presents the model for the SearchRetrieve operation and serves as a guideline for the development of *application protocol bindings* describing the capabilities and general characteristic of a server or search engine, and how it is to be accessed.

The collection includes two bindings for the SRU (Search/Retrieve via URL) protocol: SRU1.2 and SRU2.0. Both of these SRU protocols require support for CQL.

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 described in [RFC2119].

1.2 References

All references for the set of documents in this collection are supplied in the Overview document:

searchRetrieve: Part 0. Overview Version 1.0

http://docs.oasis-open.org/search-ws/searchRetrieve/v1.0/csd01/part0-overview/searchRetrieve-v1.0-csd01-part0-overview.doc

1.3 Namespace

All XML namespaces for the set of documents in this collection are supplied in the Overview document:

searchRetrieve: Part 0. Overview Version 1.0

http://docs.oasis-open.org/search-ws/searchRetrieve/v1.0/csd01/part0-overview/searchRetrieve-v1.0-csd01-part0-overview.doc

2 Model

CQL, the *Contextual Query Language*, is a formal language for representing queries to information retrieval systems. Its objective is to combine simplicity with expressiveness, to accommodate the range of complexity from very simple queries to very complex. CQL queries are intended to be human readable and writable, intuitive, and expressive.

2.1 Data Model

A server maintains a *datastore*. A unit of information in the datastore is called an *item*. The server exposes the datastore to a remote client, allowing the client to query the datastore and retrieve matching items.

2.2 Protocol Model

A CQL query is presumed to be communicated as part of a protocol message. The protocol is referred to in this document as "the search/retrieve protocol" however this standard does not prescribe any specific protocol.

Although specification of the protocol is outside the scope of CQL, the following model is assumed. There are two processing elements interfaced to one another at each of the client and server. These are referred to as (1) CQL and (2) the Protocol. At the client, CQL formulates a query and passes it to the Protocol which formulates a search/retrieve protocol request to send to the server. At the server, CQL processes the request and passes the results, including diagnostic information, to the Protocol which formulates a search/retrieve protocol response to send to the client.

2.3 Processing Model

- A client sends a search/retrieve protocol request message to a server. The request includes a CQL query and may include additional parameters to indicate how it wants the response to be composed and formatted.
- The server identifies items in the datastore that match the CQL query.
- The server sends a search/retrieve protocol response message to the client. The response includes information about the processing of the request, possibly including the query results.

2.4 Diagnostic Model

A server supplies diagnostics in the search/retrieve protocol response as appropriate. A diagnostic may be a reason why the query could not be processed, or it might be just a warning.

Diagnostics are part of the protocol and their specification is outside the scope of this standard. CQL is responsible for passing sufficient information to the Protocol so that it may generate appropriate diagnostics.

2.5 Explain Model

For any CQL implementation the server supporting that implementation provides an associated Explain record. The protocol by which the client and server communicate the CQL query and response (see Protocol Model) determines how the client accesses the Explain record from the server. (For example, for SRU, the Explain record is to be retrievable as the response of an HTTP GET at the base URL for SRU server.) The client may use the information in the Explain record to self-configure and provide an appropriate interface to the user. The Explain record provides such details as CQL context sets supported, and for each context set, indexes supported, relations, boolean operators, specification of defaults, and other detail. It also includes sample queries.

3 CQL Query Syntax: Structure and Rules

3.1 Basic Structure

A CQL query consists of either a single search clause [examples a, b], or multiple search clauses connected by boolean operators [example c]. It may have a sort specification at the end, following the 'sortBy' keyword [example d]. *Examples:*

a. cat
b. title = cat
c. .title = raven and creator = poe
d. title = raven sortBy date/ascending

3.2 Search Clause

A search clause consists of an index, relation, and a search term [example a]; or a search term alone [example b]. It must consist either of all three components (index, relation, search term) or just the search term; no other combination is allowed. If the clause consists of just a term, then the index and relation assume default values (see Context Set).
Examples:

a. title = dog
b. dog

3.3 Context Set

This section introduces context sets and describes their syntactic rules. Context sets are discussed in greater detail later.

An index is defined as part of a context set. In a CQL query the index name may be qualified by a prefix, or “short name”, indicating the context set to which the index belongs. The base index name and the prefix are separated by a dot character ('.'). (If multiple '.' characters are present, then the first should be treated as the prefix/base name delimiter.) If the prefix is not supplied, it is determined by the server.

In example (a), the qualified index name ‘dc.title’ has prefix ‘dc’ and base index name ‘title. The prefix “dc” is commonly used as the short name for the Dublin Core context set.

Context sets apply not only to indexes, but also to relations, relation modifiers and boolean modifiers (the latter two are discussed below). Conversely any index, relation, relation modifier, or boolean modifier is associated with a context set.

The prefix 'cql' is reserved for the CQL context set, which defines a set of utility (i.e. non application-specific) indexes, relations and relation modifiers. ‘cql’ is the default context set for relations, relation modifiers, and boolean modifiers. (I.e. when the prefix is omitted, ‘cql’ is assumed.) For indexes, the default context set is declared by the server in its Explain file.

As noted above, if a search clause consists of just a term [example b], then the index and relation assume default values. The term is treated as 'cql.serverChoice', and the relation is treated as '=' [example d]. Therefore examples (b) and (c) are semantically equivalent.

Each context set has a unique identifier, a URI (see Context Set URI). A server typically declares the assignment of a short name prefix to a context set in its Explain file. Alternatively, a query may include a prefix assignment [example d].

Examples:

a. dc.title = cat

b. dog

c. cql.serverChoice = dog

d. > dc = "info:srw/context-sets/1/dc-v1.1" dc.title = cat

3.4 Search Term

A search term MAY be enclosed in double quotes [example a], though it need not be [example b]. It MUST be enclosed in double quotes if it contains any of the following characters: left or right angle bracket, left or right parenthesis, equal, backslash, quote, or whitespace [example c]. The search term may be an empty string [example d].

Backslash (\) is used to escape quote (") and as well as itself.

Examples:

a. "cat"
b. cat
c. "cat dog"
d. ""

3.5 Relation

The relation in a search clause specifies the relationship between the index and search term. If no relation is supplied in a search clause, then = is assumed, which means (see CQL Context set) that the relation is determined by the server. (As is noted above, if the relation is omitted then the index MUST also be omitted; the relation is assumed to be "=" and the index is assumed to be cql.serverChoice; that is, the server chooses both the index and the relation.)

Examples*:*

a. dc.title any "fish frog"
Find records where the title (as defined by the "dc" context set) contains one of the words "fish", "frog"

b. dc.title cql.any "fish frog"
(The above two queries have the same meaning, since the default context set for relations is "cql".)

c. dc.title all "fish frog"
Find records where the title contains all of the words: "fish", "frog"

3.6 Relation Modifiers

Relations may be modified by one or more relation modifiers. Relation and modifier are separated by '/' [example a]. Relation modifiers may also have a comparison symbol and a value [examples b, c]. The comparison symbol is one of =, <, <, =, >, >=, <>. The value must obey the same rules for quoting as search terms.

A relation may have multiple modifiers, separated by '/' [example d]. Whitespace may be present on either side of a '/' character, but the relation-plus-modifiers group may not end in a '/'.

Examples:

a. title **=/relevant** cat
the relation modifier "relevant" means the server should use a relevancy algorithm for determining matches (and/or the order of the result set). When the relevant modifier is used, the actual relation ("=" in this example) is often not significant.

b. title any/**rel.algorithm=cori** cat
*This example is distinguished from the previous example in which the modifier "relevant" is from the CQL context set. In this case the modifier is "*algorithm=cori*", from the* rel *context set, in essence meaning use the relevance algorithm "cori". A description of this context set is available at http://srw.cheshire3.org/contextSets/rel/*

c. dc.title within/**locale=fr** "l m"
Find all titles between l and m, ensure that the locale is 'fr' for determining the order for what is between l and m.

d. title =***l* relevant /string** cat

3.7 Boolean Operators

Search clauses may be linked by a boolean operator **and**, **or**, **not** and **prox.**

- **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. [Example a]

- **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. [Example c]

- **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. [Example b]

- **PROX**
  'prox' is short for"proximity". The prox boolean operator allows for the relative locations of the terms to be used in order to determine the resulting set of records. [Example d]
  The set of records representing two search clauses linked by PROX is the subset, of the intersection of the two sets of records representing the two search clauses, where the locations within the records of the instances specified by the search clause bear a particular relationship to one another, the relationship specified by the prox modifiers. For example, see Boolean Modifiers in the CQL Context Set.

Boolean operators all have the same precedence; they are evaluated left-to-right. Parentheses may be used to override left-to-right evaluation [example c].
Examples:

a. dc.title = raven **and** dc.creator = poe

b. dc.title = raven **not** dc.creator = poe

c. dc.title = raven **or (**dc.creator = poe and dc.identifier = "id:1234567"**)**

d. dc.title = raven prox/unit=word/distance>3 dc.title = crow

3.8 Boolean Modifiers

Booleans may be modified by one or more boolean modifiers, separated as per relation modifiers with '/' characters. Boolean modifiers consist of a base name and may include a prefix indicating the modifier's context set [example a]. If not supplied, then the context set is 'cql'. As per relation modifiers, they may also have a comparison symbol and a value [example b] .
Examples:

a. dc.title = raven or/rel.combine=sum dc.creator = poe

b. dc.title = raven prox/unit=word/distance>3 dc.title = crow
Find records where both "raven" and "crow" are in the title, separated by at least three intervening words.

3.9 Proximity Modifiers

Basic proximity modifiers are defined in the CQL context set. Proximity units 'word', 'sentence', 'paragraph', and 'element' are defined in the CQL context set, and may also be defined in other context sets. The CQL set does not assign any meaning to these units. When defined in another context set they may be assigned specific meaning. When used in the CQL context set they should take on the meaning ascribed by some other context set, as indicated within the server's Explain file.

Thus compare "prox/unit=word" with "prox/xyz.unit=word". In the first, 'unit' is a prox modifier from the CQL set, and as such its value is server-specific. In the second, 'unit' is a prox modifier defined by the (hypothetical) xyz context set, which may assign the unit 'word' a specific meaning. The context set xyz may define additional units, for example, 'street':

```
prox/xyz.unit="street"
```

3.10 Sorting

Queries may include explicit information on how to sort the result set generated by the search.

While sorting is a function of CQL, sorting may also be a function of a search/retrieve protocol employing CQL as its query language. For example, SRU is a protocol that may employ CQL as its query language, and sorting is a function of SRU. Sorting is included as a function of CQL because it might be used with a protocol that does not support sorting. It also may be the case (as for SRU) that the protocol addresses sort only for schema elements and not search indexes. CQL addresses sort only for search indexes. When a sort specification is included in both the protocol (outside of the CQL query) and the CQL query, there is potential for ambiguity. This (CQL) standard does not attempt to address or resolve that situation. (The protocol might do so.)
The sort specification is included at the end, and is separated by a 'sortBy' keyword. The specification consists of an ordered list of indexes, potentially with modifiers, to use as keys on which to sort the result set. If multiple keys are given, then the second and subsequent keys should be used to determine the order of items that would otherwise sort together. Each index used as a sort key has the same semantics as when it is used to search.
Modifiers may be attached to the index in the same way as to booleans and relations in the main part of the query. These modifiers may be part of any context set, but the CQL context set and the Sort Context Set are particularly important.

Note that modifiers may be attached to indexes only in a sort clause. Modifiers may not be attached to indexes in a search clause.

Examples:

a. cat sortBy dc.title

b. dinosaur sortBy dc.date/sort.descending dc.title/sort.ascending

3.11 Case Sensitivity

All parts of CQL are case insensitive apart from user supplied search terms, values for modifiers, and prefix map identifiers, which may or may not be case sensitive.

4 CQL Query Syntax: ABNF

Following is the Augmented Backus-Naur Form (ABNF) definition for CQL. ABNF is specified in RFC 5234 (STD 68).

The equals sign ("=") separates the rule name from its definition elements, the forward slash ("/") separates alternative elements, square brackets ("[", "]") around an element list indicate an optional occurrence, while variable repetition is indicated by an asterisk ("*") preceding an element list with parentheses ('(", ")") used for grouping elements.

```
; A. Query

            cql-query  =  query [sort-spec]

; B. Search Clauses

                query  =  *prefix-assignment search-clause-group

  search-clause-group  =  search-clause-group boolean-modified subquery |
                          subquery

             subquery  =  "(" query ")" / search-clause

        search-clause  =  [index relation-modified] search-term

          search-term  =  simple-string / quoted-string / reserved-string

; C. Sort Spec

            sort-spec  =  sort-by 1*index-modified

              sort-by  =  "sortby"

; D. Prefix Assignment

     prefix-assignment  =  ">" [prefix "="] uri

               prefix   =  simple-name

                  uri   =  quoted-uri-string

; E. Indexes

       index-modified   =  index [modifier-list]

                index   =  simple-name / prefix-name

; F. Relations

    relation-modified   =  relation [modifier-list]

             relation   =  relation-name / relation-symbol

        relation-name   =  simple-name / prefix-name
```

```
relation-symbol     =   "=" / ">" / "<" / ">=" / "<=" / "<>" / "=="

; G. Booleans

boolean-modified    =   boolean [modifier-list]

boolean             =   "and" / "or" / "not" / "prox"

; H. Modifiers

modifier-list       =   1*modifier

modifier            =   "/" modifier-name [modifier-relation]

modifier-name       =   simple-name

modifier-relation   =   relation-symbol modifier-value

modifier-value      =   simple-string / quoted-string

; I. Terminal Aliases

prefix-name         =   prefix "." simple-name
                        ; Prefix (simple-name) and name (simple-name)
                        separated
                        ; by dot character (".").
                        ;
                        ; No whitespace allowed before or after the dot character
                        ; (".")

quoted-uri-string   =   ; Double quotes enclosing a URI string.
                        ;
                        ; RFC 3986 (STD 66) specifies the allowed characters
                        ; for a URI which all fall within the printable subset of
                        ; US-ASCII.

reserved-string     =   boolean / sort-by

simple-name         =   simple-string

; J. Terminals

quoted-string       =   ; Double quotes enclosing a sequence of any characters
                        ; except double quote unless preceded by a backslash
                        ; character ("\").
                        ;
                        ; Backslash escapes the character following it. The
                        ; surrounding double quotes are not included in the
                        value.

simple-string       =   ; Any sequence of non-whitespace characters that does
                        not
                        ; include any of the following graphic characters:
                        :
                        ;     " ( ) / < = >
```


5 Context Sets

CQL is so-named ("Contextual Query Language") because it is founded on the concept of searching by semantics and context, rather than by syntax. CQL uses context sets to provide the means to define community-specific semantics. Context sets allow CQL to be used by communities in ways that the designers could not have foreseen, while still maintaining the same rules for parsing.

A context set defines one or more of the following constructs:

- Indexes
- Relations
- Relation modifiers
- Boolean modifiers
- Index modifiers (for use in a sortBy clause)

Each occurrence of one of these constructs in a CQL query belongs to a context set, implicitly or explicitly. There are rules to determine the prevailing default set if it is not explicitly indicated.

For example:

- In the search clause:
  dc.title any/rel.algorithm=cori cat
  - The **index**, 'title', belong to the context set 'dc'. More accurately, it belongs to the context set whose short name is "dc"; in most cases this will be the Dublin Core context set as 'dc' is its conventional short name. Every context set has a (permanent) URI and a short name which may vary from query to query. The association of a short name to a context set is discussed below.
  - The **relation, '**any**',** belongs to the cql context set.
  - The **relation modifier**, rel.algorithm, belongs to the context set whose short name is 'rel'.
- In the boolean triple:
  dc.title = raven or/rel.combine=sum dc.creator = poe
  - The **boolean modifier,** 'rel.combine=sum' **(**modifying the boolean operator 'or') belongs to the context set whose short name is 'rel'.
- In the query
  dc.creator=plews sortby dc.title/sort.respectCase
  - The **index modifier**, 'sort.respectCase' (modifying the index dc.title in the sort clause) belongs to the context set whose short name is 'sort' (presumably the Sort Context Set.)

5.1 Context Set URI

As noted above each context set has a unique identifier, a URI. It may, but need not, be an 'http:' URI. It might be an 'info:' URI. For example, the CQL Context Set is identified by the URI

```
info:srw/cql-context-set/1/cql-v1.2
```

There is a list of several useful context sets at http://www.loc.gov/standards/sru/resources/context-sets.html.
Note that among the identifying URIs, some are 'http:' URIs and others are 'info:' URIs; any other appropriate URI scheme may be used. However this standard provides a means for an implementor to

register an “info:srw” subspace, where context set (and other object) URIs may be registered. See http://www.loc.gov/standards/sru/resources/infoURI.html.

5.2 Context Set Short Name

As noted above, within a CQL query, a context set is denoted by a prefix, which is a short name for the context set. The association of the short name to the context set may be assigned in the server’s Explain file, or within the CQL query. For example, in the query:
> dc = "info:srw/context-sets/1/dc-v1.1" dc.title = cat

‘> dc = "info:srw/context-sets/1/dc-v1.1"‘ associates the short name ‘dc’ to the URI info:srw/context-sets/1/dc-v1.1 (which identifies the Dublin Core context set) so that ‘dc’ may be used subsequently within the query as the prefix identifying that context set. Note that the assignment if a short name to a URI does not persist across queries, regardless of what protocol is used.

5.3 Defining a Context Set

Anyone can define a context set, all that is required is a URI (as described above in Context Set URI) to identify it. The definition should list the URI, the preferred short name, and all indexes, relations, relation modifiers, boolean modifiers, and index modifiers (used in sort clauses) defined by the context set.

A context set may define any or all of these constructs. If one wants to define a single relation (no indexes, modifiers, etc.) a new context set may be defined for just that single relation. Many context sets likely will define indexes only.

5.4 Standardization and Registration of Context Sets

Some context sets will be standardized, some will be registered (whether standardized or not) and some will be neither standardized nor registered.

5.4.1 Standard Context Sets

5.4.2 Core Context Sets

The CQL standard includes as normative (and therefore standardizes) definitions for three context sets considered essential to the use of CQL. These are the CQL Context Set , the Sort Context Set, and the Dublin Core Context Set. They are defined in the first three annexes.

5.4.2.1 Standard Application Context Sets

Any individual or community that defines a context set may choose to standardize it within an appropriate standard body. The decision whether or not to standardize it, and in what standards body, is outside the scope of this standard.
An example of an application context set is the Bibliographic Context Set, which is included as a non-normative annex. (It is included as an example.) It is not currently a formal standard but may be standardized (by some standards body) in the future.

5.4.3 Registered Context Sets

The CQL Maintenance Agency provides a register of context sets. Any individual or community that defines a context set may request that it be registered. The current registry is at http://www.loc.gov/standards/sru/resources/context-sets.html. Registration is a service provided to facilitate discovery of context sets by developers and users.
Registration and standardization are independent. A context set may be standardized and registered, standardized and not registered, registered and not standardized, or neither standardized nor registered.

6 Conformance

6.1 Client Conformance

Three levels of support are defined for a CQL client. In order for a client to claim conformance to CQL it must support at least level 0:

6.1.1 Level 0

The client must be able to form a term-only query.
*Note: The term is either a single word, or, if multiple words separated by spaces then the entire search term is quoted. If the term includes quote marks, they must be escaped by preceding them with a backslash, e.g."*raising the \"titanic\""*.)*

6.1.2 Level 1

1. Support Level 0.
2. Be able to form at least one of :
   (a) a search clause consisting of 'index relation searchTerm';
   (b) queries where search terms are combined with booleans, e.g. "term 1 AND term2"

   Note: (b) does not require support for queries of the form:
   index relation term1 AND index relation term2
   It requires support for queries where the search clauses are term-only (do not include index or relation).

6.1.3 Level 2

The client must:

1. Support Level 1.
2. Be able *to* formulate all queries described in this standard, including those described by the CQL context set.

6.2 Server Conformance

Three levels of support are defined for a CQL server. In order for a server to claim conformance to CQL it must support at least level 0:

6.2.1 Level 0

The server must:

1. Be able to process a term-only query. (See Client Conformance, Level 0.)
2. Be able to inform the Protocol that the query is not supported, in the event of any unsupported query.
   Note: The intent is that the protocol will issue a diagnostic from server to client. However this is beyond the scope of the CQL standard.

6.2.2 Level 1

The server must:

1. Support Level 0.
2. Be able to *parse* both:
   (a) search clauses consisting of 'index relation searchTerm'; and
   (b) queries where search terms are combined with booleans, e.g. "term 1 AND term2"
3. *Support* at least one of (a) and (b).

Notes

1. *In 2 and 3:*

   i. *"parse both" mean that the server must at minimum be able to recognize (a) search clauses consisting of 'index relation searchTerm' or (b) queries where search terms are combined with booleans, even if it does not support it, and be able to inform the Protocol so that it may convey an appropriate diagnostic.*
   ii. *"Support" means that it must be able to process - not just be able to parse - at least one.*

2. *(b) does not require ability to parse or support queries such as*: index relation term1 AND index relation term2 *but rather queries where the search clauses are terms-only (do not include index or relation).*

6.2.3 Level 2

The server must:

1. Support Level 1.
2. Be able to *parse* all of CQL and respond with appropriate error messages to the search/retrieve protocol interface.

Note: (2) does not require support *for all of CQL, but rather that the server be able to* parse *all of CQL.*

Appendix A. Acknowledgments

Acknowlegements are supplied in the Overview document:

searchRetrieve: Part 0. Overview Version 1.0

http://docs.oasis-open.org/search-ws/searchRetrieve/v1.0/csd01/part0-overview/searchRetrieve-v1.0-csd01-part0-overview.doc

Appendix B. The CQL Context Set

Normative Annex

The CQL context set defines a set of indexes, relations and relation modifiers. The indexes defined are utility indexes, generally useful across applications. These utility indexes are for instances when CQL is required to express a concept not directly related to the data, or for indexes applicable in most contexts.

The reserved name for this context set is: **cql**

The identifier for this context set is: info:srw/cql-context-set/1/cql-v2.0

B.1 Indexes

- **serverChoice**
  This is the default when the index and relation is omitted from a search clause. 'cql.serverChoice' means that the server will choose one or more indexes in which to search for the given term. The relation used is '=', hence 'cql.serverChoice="term"' is an equivalent search clause to '"term"'.

! **resultSetId**

Note: Discussion of the resultSetId index assumes that CQL is being used with a protocol that declares a result set model for example, the SRU protocol.

A result set id may be used as the index in a search clause [example a]. This is a special case, where the index and relation are expressed as "cql.resultSetId =" and the term is a result set id that has been previously returned by the server, for example in the 'resultSetId' element of an SRU response. It may be used by itself in a query to refer to an existing result set from which records are desired. It may be used to create a new result set via manipulation of existing result sets [example b]. It may also be used to restrict a query to a given result set. in conjunction with other resultSetId clauses or other indexes, combined by boolean operators. The semantics when resultSetId is used with relations other than "=" is undefined.

Examples:

a. cql.resultSetId = "5940824f-a2ae-41d0-99af-9a20bc4047b1"
Match all records in the result set with the given identifier.

b. cql.resultSetId = "a" AND cql.resultSetId = "b"
Create a new result set which is the intersection of these two result sets.

c. cql.resultSetId = "a" AND dc.title=cat
Apply the query 'dc.title=cat' to result set "a".

! **allRecords**

A special index which matches every record available. Every record is matched no matter what values are provided for the relation and term, but the recommended syntax is: cql.allRecords = 1

Example:

" cql.allRecords = 1 NOT dc.title = dog
Search for all records that do not match 'dog' as a word in title.

- **allIndexes**
  The 'allIndexes' index will result in a search equivalent to searching all of the indexes (in all of the context sets) that the server has access to. AllIndexes is not equivalent to a full-text search: not all content is necessarily indexed, and content not indexed would not be searchable with the allIndexes index.

Examples:

- cql.allIndexes = dog
  If the server had three indexes title, creator, and date, then this would be the same as title = dog or creator = dog or date = dog

B.2 Relations

B.2.1 Implicit Relations

These relations are defined as such in the grammar of CQL. The cql context set only defines their meaning, rather than their existence.

- =
  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. If the term is numeric, the most commonly chosen relation is '==='. For a string term, either 'adj' or '==' as appropriate for the index and term. The Explain file lists for every combination of index and term what relation is used when '=' is supplied.

Examples:

- animal.numberOfLegs = 4
  Recommended to use '=='

- dc.identifer = "gb 141 staff a-m"
  Recommended to use '=='

- dc.title = "lord of the flies"
  Recommended to use 'adj'

- dc.date = "2004 2006"
  Recommended to use 'within'

- ==
  This relation is used for exact equality matching. The term in the data is exactly equal to the term in the search. A relation modifier may be included to specify how whitespace (trailing, preceding, or embedded) is to be treated (for example, the CQL relation modifier 'honorWhitespace').

Examples:

- dc.identifier == "gb 141 staff a-m"
  Search for the string 'gb 141 staff a-m' in the identifier index.

- dc.date == "2006-09-01 12:00:00"
  Search for the given datestamp.

- animal.numberOfLegs == 4
  Search for animals with exactly 4 legs.

- <>
  This relation means 'not equal to' and matches anything which is not exactly equal to the search term.

Examples:

" dc.date <> 2004-01-01
Search for any date except the first of January, 2004

" dc.identifier <> ""
Search for any identifier which is not the empty string.

- <, >, <=,>=
  These relations retain their regular meanings as pertaining to ordered terms (less than, greater than, less than or equal to, greater than or equal to).

Examples:

" dc.date > 2006-09-01
Search for dates after the 1st of September, 2006

" animal.numberOfLegs < 4
Search for animals with less than 4 legs.

B.2.2 Defined Relations

These relations are defined as being widely useful as part of a default context set.

- adj
  Adjacency. Used for phrase searches. All of the words in the search term must appear, and must be adjacent to each other in the record in the order of the search term. The adj relationship has an implicit relation modifier of 'cql.word', which may be changed by use of alternative relation modifiers.
  An adjacency query could also be expressed using the PROX boolean operator, for example,
  title adj “a b c”
  would be equivalent to
  (title=a prox/distance=1/ordered title=b) prox/distance=1/ordered title=c
  The space character is the default delimiter to be used to separate words in the search term for the ‘adj’ relation. A different delimiter may be specified in the server’s Explain file.

Examples*:*

" dc.title adj "lord of the flies"
Search for the phrase 'lord of the flies' somewhere in the title.

" dc.description adj "blue shirt"
Search for 'blue' immediately followed by 'shirt' in the description.

- all, any
  These relations may be used when the term contains multiple items to indicate "all of these items" or "any of these items". These queries could be expressed using boolean AND and OR respectively. These relations have an implicit relation modifier of 'cql.word', which may be changed by use of alternative relation modifiers. Relation ‘all’ may be used with relation modifier ‘windowSize’ to further require that the words all occur within a window of specified size.

Examples:

" dc.title all "lord flies"
Search for both lord and flies in the title.

" dc.title all/windowSize=6 "cat hat rat"
Find "cat", "hat", and "rat" within a 6-word window.

" dc.description any "computer calculator"
Search for either computer or calculator in the description.

! within
Within may be used with a search term that has multiple dimensions.(Dimension values are delimited by space.) It matches if the database's term falls completely within the range, area or volume described by the search term, inclusive of the extents given.

Examples*:*

" dc.date within "2002 2003"
Search for dates between 2002 and 2003 inclusive.

" animal.numberOfLegs within "2 5"
Search for animals that have 2,3,4 or 5 legs.

! encloses
R*oughly the opposite of within and similarly is used when the index's data has multiple dimensions. It matches if the database's term fully encloses the search term.*

Examples:

" "geo.dateRange encloses 2002
Search for ranges of dates that include the year 2002.

" geo.area encloses "45.3 19.0"
Search for any area that encloses the point 45.3, 19.0

B.3 Relation Modifiers

B.3.1 Functional Modifiers

! **relevant**
The server should use a relevancy algorithm for determining matches and the order of the result set.

! **fuzzy**
The server should be liberal in what it counts as a match. The exact details of this are left up to the server, but might include permutations of character order, off-by-one for numerical terms and so forth.

! **partial**
When used with within or encloses, there may be some section which extends outside of the term. This permits for the database term to be partially enclosed, or fall partially within the search term.

! **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). This modifier may be used in sort keys to ensure that

terms with the same letters in different cases are sorted together or separately, respectively. These modifiers may be used in sort keys.

- **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). This modifier may be used in sort keys, to ensure that characters with diacritics are sorted together or separately from those without them. These modifiers may be used in sort keys.

- **locale=value**
  The term should be treated as being from the specified locale. Locales are identifiers for a grouped specification of options in relation to sort order (collation), names for time zones, languages, countries, scripts, measurement units, numbers and other elements. Values for locales can be found in the Unicode Common Locale Data Repository (CLDR) http://unicode.org/cldr/ *which points to* http://www.iana.org/assignments/language-subtag-registry *. 2 character language codes are specified, e.g. "es" is Spanish, "en" is English. Specifically in relation to sort order, locales indicate how data is normalized, e.g. whether sort order is case-sensitive or insensitive and how characters with diacritics are normalized. The language code may be modified by a 2 character country code as per ISO 3166, e.g. "en-UK" and "en-US" The default locale is determined by the server. As well as being used in a query, locales may be specified in sort keys.*

- **windowSize=value**
  Used with relation 'all', to specify that a set of words (two or more) are contained within a span of a specified number of words.

- **Weight=value**
  Specifies a weight to be assigned to this search clause, relative to other search clauses. A positive integer, default value is 1.

Examples:

- person.phoneNumber =/fuzzy "0151 795-4252"
  Search for a phone number which is something similar to '0151 795-4252' but not necessarily exactly that.

- "fish" sortBy dc.title/ignoreCase
  Search for 'fish', and then sort the results by title, case insenstively.

- dc.title within/locale=fr "l m"
  Find all titles between l and m, ensure that the locale is 'fr' for determining the order for what is between l and m.

- dc.title all/windowSize=6 "cat hat rat"
  Find "cat", "hat", and "rat" within a 6-word window.

B.3.2 Term-format Modifiers

These modifiers specify the format of the search term to ensure that the correct comparison is performed by the server. These modifiers may all be used in sort keys.

- **word**
  The term should be broken into words, according to the server's definition of a 'word'.

- **string**
  The term is a single item, and should not be broken up.

598 ! **isoDate**
599 *Each item within the term conforms to the ISO 8601 specification for expressing dates.*

600 ! **number**
601 *Each item within the term is a number.*

602 ! **uri**
603 *Each item within the term is a URI.*

604 ! **oid**
605 *Each item within the term is an ISO object identifier, dot-separated format.*

606 *Examples:*

607 ! dc.title =/string “today’s winners and today’s losers ”
608 *Search in title for the term as a string', rather than as a sequence of words. (Equivalent to the*
609 *use of == as the relation)*

610 ! zeerex.set ==/oid "1.2.840.10003.3.1"
611 *Search for the given OID as an attribute set.*

612 ! squirrel sortby numberOfLegs/number
613 *Search for squirrel, and sort by the numberOfLegs index ensuring that it is treated as a number,*
614 *not a string. (eg '2' would sort after '10' as a string, but before it as a number.)*

615 ### B.3.3 Matching

616 ! **masked** (default modifier)
617 The following masking rules and special characters apply for search terms, unless overridden in
618 a profile via a relation modifier. To explicitly request this functionality, add 'cql.masked' as a
619 relation modifier.

620 " A single asterisk (*) is used to mask zero or more characters.

621 " A single question mark (?) is used to mask a single character, thus N consecutive
622 question-marks means mask N characters.

623 " Carat/hat (^) is used as an anchor character for terms that are word lists, that is, where
624 the relation is 'all' or 'any', or 'adj'. It may not be used to anchor a string, that is, when the
625 relation is '==' (string matches are, by default, anchored). It may occur at the beginning or
626 end of a word (with no intervening space) to mean right or left anchored."^" has no
627 special meaning when it occurs within a word (not at the beginning or end) or string but
628 must be escaped nevertheless.

629 " Backslash (\) is used to escape '*', '?', quote (") and '^' , as well as itself. Backslash not
630 followed immediately by one of these characters is an error.

631 Examples:

632 " dc.title = c*t
633 *Matches words that start with c and end in t*

634 " dc.title adj "*fish food*"
635 *Matches a word that ends in fish, followed by a word that starts with food.*

636 " dc.title = c?t

Matches a three letter word that starts with c and ends in t.

" dc.title adj "^cat in the hat"
Matches 'cat in the hat' where it is at the beginning of the field

" dc.title any "^cat ^dog rat^"
Matches a string with 'cat' or 'dog' at the beginning or 'rat' at then end: 'cat eats rat', 'dog eats rat', but not 'rat eats cat'.

" dc.title == "\"Of Couse\", she said"
Escape internal double quotes within the term.

! **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 ir not to honor extraneous whitespace.

! **Substring**
The 'substring' modifier may be used to specify a range of characters (first and last character) indicating the desired substring within the field to be searched. The modifier takes a value, of the form "start:end" where start and end obey the following rules:

" *Positive integers count forwards through the string, starting at 1. The first character is 1, the tenth character is 10.*

" *Negative integers count backwards through the string, with -1 being the last character.*

" *Both start and end are inclusive of that character.*

" *If omitted, start defaults to 1 and end defaults to -1.*

Examples:

" *marc.008 =/substring="1:6" 920102*

" *dc.title =/substring=":" "The entire title"*

" *dc.title =/substring="2:2" h*

" *dc.title =/substring="-5:" title*

! **regexp**
The term should be treated as a regular expression. Any features beyond those found in modern POSIX regular expressions are considered to be server dependent. This modifier overrides the default 'masked' modifier, above. It may be used in either a string or word context.

Examples:

" dc.title adj/regexp "(lord|king|ruler) of th[ea] r.*s"
Match lord or king or ruler, followed by of, followed by the or tha, followed by r plus zero or more characters plus s.

B.4 Boolean Modifiers

The CQL context set defines the following boolean modifiers, which are only used with the prox boolean operator.

- distance symbol value
  The distance that the two terms should be separated by.

  - " *Symbol is one of: < > <= >= = <>*
    If the modifier is not supplied, it defaults to <=.

  - " *Value is a non-negative integer.*
    If the modifier is not supplied, it defaults to 1 when unit=word, or 0 for all other units.

- container=containerName
  A container is a structure containing one or more indexes. For example the server may support a container whose name is 'author' that contains indexes 'name' and 'date'. In that case the server would support a query (see example) to find an author with a specific name and date. (This is contrasted with a boolean query which may return undesired results because they have multiple authors, some of which have the desired name but the wrong date and others the specified date but the wrong name.) The server should list supported containers in its Explain file, and for each container, the indexes that it contains.

- unit=value
  The type of unit for the distance.
  Value is one of: 'paragraph', 'sentence', 'word' and 'element', and defaults to 'word'. These values are explicitly undefined. They are subject to interpretation by the server. See Proximity Units .

- unordered
  The order of the two terms is unimportant. This is the default.

- ordered
  The order of the two terms must be as per the query.

Examples:

- cat prox/unit=word/distance>2/ordered hat
  Find 'cat' where it appears more than two words before 'hat'

- cat prox/unit=paragraph hat
  Find cat and hat appearing in the same paragraph (distance defaulting to 0) in either order (unordered default)

- name=jones prox/container=author date=1950
  Find the name 'jones' and date '1950' in the same author field.

- jack PROX/container=author jones
  Find 'jack' and 'jones' within the same author field. (In this example, both 'jack' and 'jones' assume the default relation and index for the server, and that index is assumed to be supported for the container 'author'.)

- jack PROX/container=author/distance<=2/ordered jones
  Find 'jack' followed by 'jones' within the same author field,separated by two words or less

B.4.1 Proximity Units

As noted above, proximity units 'paragraph', 'sentence', 'word' and 'element' are explicitly undefined when used by the CQL context set. Other context sets may assign them specific values.

Thus compare "prox/unit=word" with "prox/xyz.unit=word". In the first, 'unit' is a prox modifier from the CQL set, and as such its values are undefined, so 'word' is subject to interpretation by the server. In the second, 'unit' is a prox modifier defined by the xyz context set, which may assign the unit 'word' a specific meaning.

Other context sets may define additional units, for example, 'street': ‘prox/xyz.unit="street"

‘

Appendix C. The Sort Context Set

Normative Annex

The *sort context set* defines a set of index modifiers to be used within a sortby clause.

The URI for this context set is: **info:srw/cql-context-set/1/sort-v1.0**

The recommended short name is: **sort**

CQL does not permit index modifiers, except within a sort clause. For example in the CQL query: "*author=wolfe sortby title"* 'sortby title' is a sort clause; 'title' is an index. 'author', which is the primary index of query, may not have a modifier, but 'title', which is the index of the sort clause, may.

Thus for example, in the CQL query: "*author=wolfe sortby title/ascending"* 'ascending' is an index modifier.

The sort context set defines index modifiers only. It does not define any of the other constructs of context sets (indexes, relations, relation modifiers, relation qualifiers, or boolean modifiers). The index modifiers defined by the sort context set are as shown in the following table.

Modifier	Description
ignoreCase	Case-insensitive sorting: for example, unit and UNIT sort together.
respectCase	Case-sensitive sorting: for example, unit and UNIT sort separately.
ignoreAccents	Accent-insensitive sorting: for example sorensen and sørensen sort together.
respectAccents	Accent-sensitive sorting: for example sorensen and sørensen sort separately.
ascending	Sort in ascending order.
descending	Sort in descending order.
missingOmit	Records that have no value for the specified index are omitted from the sorted result set.
missingFail	Records that have no value for the specified index cause the search/sort operation to fail.
missingLow	Records that have no value for the specified index are treated as if they had the lowest possible value (they sort first in ascending order and last in descending order).
missingHigh	Records that have no value for the specified index are treated as if they had the highest possible value.
missingValue =*value*	Records that have no value for the specified index are treated as if they had the specified *value*.
Locale =*value*	Sort according to the specified locale, which will in general include specifications for whether sorting is case-sensitive or insensitive, how it treats accents, etc. The *value* is usually of the form C, french, fr_CH, fr_CH.iso88591 or similar.
unicodeCollate =*value*	Specfies the Unicode collation level. The *value* should be a small integer as described in the *Unicode Collation Algorithm* report at www.unicode.org/reports/tr10

C.1 Examples

- dc.creator=plews sortby dc.title/sort.respectCase
  Sort by title, case sensitive

- dc.creator=plews sortby dc.title/sort.respectCase/sort.descending
  Sort case sensitive and in descending order

- dc.creator=plews sortby dc.date/sort.missingOmit
  Sort by date: records that have no date field are omitted from the result set.

- dc.creator=plews sortby dc.date/sort.missingValue=1970
  Sort by date: records that have no date field are sorted as though they had a date of 1970

Appendix D. The Dublin Core Context Set

Normative Annex

The *Dublin Core context set* defines 15 indexes, corresponding to the 15 Dublin Core (simple) elements.

The URI for this context set is: **info:srw/cql-context-set/1/dc-v1.1**

The recommended short name is: **dc**

D.1 Indexes

1. title
2. creator
3. subject
4. description
5. publisher
6. contributor
7. date
8. type
9. format
10. identifier
11. source
12. language
13. relation
14. coverage
15. rights

The semantics of these indexes are the same as those of the corresponding Dublin Core elements. See sections 4.1-4.15 of http://dublincore.org/documents/usageguide/elements.shtml.

D.2 Relations

No relations are defined for this context set.

D.3 Relation Modifiers

No relation modifiersare defined for this context set.

D.4 Boolean Modifiers

No boolean modifiers are defined for this context set.

Appendix E. Bib Context Set

Non-normative Annex

The *bib context set* defines bibliographic indexes and modifiers.

The indexes and modifiers are based on MODS, i.e. MODS is used for reference semantics; this does not presume that the data being searched is MODS.

- **URI for this context set**: info:srw/cql-context-set/1/bib-v1
- **Recommended short name**: bib

Examples of the use of this context set are supplied in the non-normative Annex Bibliographic Searching Examples.

E.1 Indexes

E.1.1 Title Indexes

Note that this context set does not define an index for "title proper"; dc.title may be used.

- bib.titleAbbreviated
- bib.titleUniform
- bib.titleTranslated
- bib.titleAlternative
- bib.titleSeries

E.1.2 Name Indexes

- bib.name
- bib.namePersonal
- bib.namePersonalFamily
- bib.namePersonalGiven
- bib.nameCorporate
- bib.nameConference

E.1.3 Subject Indexes

- bib.subjectPlace
- bib.subjectTitle
- bib.subjectName
- bib.subjectOccupation

E.1.4 Date Indexes

- bib.dateIssued
- bib.dateCreated
- bib.dateValid
- bib.dateModified

- bib.dateCopyright

E.1.5 Part Indexes

- bib.volume
- bib.issue
- bib.startPage
- bib.endPage

E.1.6 Additional Indexes

- *genre*: bib.genre
- *Audience*: bib.audience
- *Classification*: bib.classification
- *Place of Origin*: bib.originPlace
- *Edition*: bib.edition
- *Issuance*: bib.issuance
  Values:
  - continuing
  - monograph

E.2 Relations

No relations are defined for this context set.

E.3 Relation Modifiers

E.3.1 Relation Modifiers for title indexes

- bib.portion
  Values:
  - main
  - sub
  - partNum
  - partName

- bib.titleAuthority (for titleUniform only)
  Values:
  - lcnaf

E.3.2 Relation Modifiers for title indexes

- bib.date
- bib.nameAuthority
- bib.role
- bib.roleAuthority (**default** marcrelator)

E.3.3 Relation Modifiers for subject indexes

- bib.subjectAuthority (e.g. 'marcgac', 'marccountry', 'iso3166', 'lcsh', 'lcnaf')

E.3.4 Relation Modifiers for identifier indexes

Note that this context set does not define indexes for identifiers. These modifiers may be used for example with dc.identifier.

- bib.identifierAuthority
  Values:

Among the values for this modifier are the following initial set.

  - hdl
  - doi
  - isbn
  - isrc
  - ismn
  - issn
  - local
  - lccn
  - stock-number
  - uri

These are represented, respectively by the following URIs:

  - info:/srw/1/vocabulary/identifierType/hdl
  - info:/srw/1/vocabulary/identifierType/doi
  - info:/srw/1/vocabulary/identifierType/isbn
  - info:/srw/1/vocabulary/identifierType/isrc
  - info:/srw/1/vocabulary/identifierType/ismn
  - info:/srw/1/vocabulary/identifierType/issn
  - info:/srw/1/vocabulary/identifierType/local
  - info:/srw/1/vocabulary/identifierType/lccn
  - info:/srw/1/vocabulary/identifierType/stock-number
  - info:/srw/1/vocabulary/identifierType/uri

For these values, the actual parameter value used may be the URI or it may be the term itself. The rule is that whenever the parameter value does not take the form of a URI, then it is assumed to be prefixed by the string 'info:srw/resultCountPrecision/1/'.
In these URIs, the path component '1' is the authority component; '1' refers to the SRU Maintenance Agency. Other authorities will be registered upon request. See http://www.loc.gov/standards/sru/resources/infoURI.html for details. In this manner additional values may be defined. The 'info' URI mechanism is not intended to preclude use of other types of URIs to represent values of this parameter.

E.3.5 Relation Modifiers for date indexes

- bib.dateAuthority
  Values:
  - w3cdtf (see http://www.w3.org/TR/NOTE-datetime).
  - edtf (see http://www.loc.gov/standards/datetime/). This is the default value.

E.3.6 Relation Modifiers for format index

- bib.formatAuthority

E.3.7 Relation Modifiers for genre index

- bib.genreAuthority
  Values:
  - modsGenre (default)

E.3.8 Relation Modifiers for type indexes

Note that this context set does not define indexes for type. These modifiers may be used for example with dc.type.

- bib.typeAuthority
  Values:
  - modsResource

E.3.9 Relation Modifiers for target audience index

- bib.audienceAuthority
  Values:
  - modsAudience (See http://www.loc.gov/marc/sourcecode/target/targetlist.html). This is the default value.

E.3.10 Relation Modifiers for classification index

- bib.classAuthority
  Values:
  - lcc (This is the default value.)

E.3.11 Relation Modifiers for Place of Origin index

- bib.geoUnit
  Values:
  - country
  - city
- bib.placeAuthority
  Values:
- marcCC (country code)
- marcCN (country name)

See http://www.loc.gov/marc/countries/

E.3.12 Relation Modifiers for language indexes

Note that this context set does not define indexes for language. These modifiers may be used for example with dc.language.

- bib.languageAuthority
  Values:
  - rfc3066
  - iso639-2b
    Default is server defined

E.4 Relation Qualifiers

No relation qualifiers are defined for this context set.

E.5 Boolean Modifiers

No boolean modifiers are defined for this context set.

E.6 Summary Table

Category	Indexes	Modifiers
Title	• bib.titleAbbreviated • bib.titleUniform • bib.titleTranslated • bib.titleAlternative • bib.titleSeries	• bib.portion (main, sub, partNum, partName) • bib.titleAuthority (for titleUniform only)
Name	• bib.name • bib.namePersonal • bib.namePersonalFamily • bib.namePersonalGiven • bib.nameCorporate • bib.nameConference	• bib.date • bib.nameAuthority • bib.role • bib.roleAuthority default marcrelator
Subject	• bib.subjectPlace • bib.subjectTitle • bib.subjectName • bib.subjectOccupation	• bib.subjectAuthority
Identifier		• bib.identifierAuthority

Date	• bib.dateIssued
	• bib.dateCreated
	• bib.dateValid
	• bib.dateModified
	• bib.dateCopyright
		• bib.dateAuthority
		  ○ edtf
		  ○ w3cdtf

Resource Type		• bib.typeAuthority
Format		• bib.formatAuthority
Genre	• bib.genre	• bib.genreAuthority
Target Audience	• bib.audience	• bib.audienceAuthority
Classification	• bib.classification	• bib.classAuthority

Place of Origin	• bib.originPlace	• bib.geoUnit • bib.placeAuthority
Language		• bib.languageAuthority Default: server defined
Edition	• bib.edition	
Part	• bib.volume • bib.issue • bib.startPage • bib.endPage	
Issuance	• bib.issuance	

E.7 Bibliographic Searching Examples

E.7.1 Examples of Searching by Title

1. bib.titleUniform=/bib.portion=main/bib.titleAuthority=lcnaf "Symphonies, no. 5, op. 67, C minor"
2. bib.titleTranslated=/bib.portion=main/lang=fr "homme qui voulut être roi"
3. dc.title="Annual report of notifiable diseases"
4. dc.title="Annual report of notifiable diseases" OR bib.titleAbbreviated="Annu. rep. notif. dis."
5. dc.title=/lang=rus "Geodezja i urzadzenia roline" OR bib.titleTranslated=/lang=eng "Land surveying and agriculture equipment"
6. dc.title="Focus on grammar" AND bib.titleSub="basic level"

Notes:

- As seen in these examples there is no general 'title' index defined for the bib set. To search on unqualified 'title', for example to search for a list of words anywhere within a title field, dc.title is to be used.
- Similarly there is no bib.titleProper index defined. To search on "title proper" dc.title is to be used.

E.7.2 Examples of Searching by Name

1. bib.namePersonal="Herb Plews"
2. bib.namePersonalGiven=herb PROX bib.namePersonalFamily=plews
3. bib.namePersonal=/bib.role=shortstop "Herb Plews"
4. bib.nameCorporate=ibm
5. bib.nameConference="International Workshop on Plasma-Based Ion Implantation 1993 : University of Wisconsin--Madison"
6. bib.NamePersonal=/bib.nameAuthority=lcnaf/bib.role=composer/bib.roleAuthority=marcrelator "Beethoven, Ludwig van, 1770-1827"
7. bib.NamePersonal=/bib.role=author/bib.roleAuthority=marcrelator "George Orwell"
8. bib.namePersonal=/bib.date="1835-1913" "Albert Babeau"
9. dc.contributor="Florida Department of Agriculture and Consumer Affairs"

Notes:

- In example 6, "role=composer/bib.roleAuthority=marcrelator" means that the 'role' "composer" is taken from the list 'marcrelator' which is intended to be a nickname for the list of roles at http://www.loc.gov/marc/sourcecode/relator/relatorlist.html.
- So, as seen in example 7, to do an author search, use "bib.role=author/bib.roleAuthority=marcrelator".
- lcnaf refers to the LC name authority file, searchable at http://authorities.loc.gov/. The authorized name heading, "Beethoven, Ludwig van, 1770-1827" (in example 5), can be found there.
- To search by contributor use dc.contributor; to search by publisher, use dc.publisher; to search by creator, use dc.creator. That is, use these instead of role=contributor, role=publisher or role=creator.

E.7.3 Examples of Searching by Subject

1. dc.subject="Food additives -- Law and legislation"
2. dc.subject=/bib.subjectAuthority=lcsh "Food additives -- Law and legislation"
3. bib.subjectName= "Ted Williams"
4. bib.subjectName=/bib.subjectAuthority=lcnaf "Williams, Ted, 1918-2002"

Notes:

- No bib index is defined to search on unqualified 'subject', instead (as seen in example 1) dc.subject should be used.
- Similarly there is no bib.subjectTopic index defined. To search on "subject - topic" dc.subject is to be used.

E.7.4 Examples of Searching by Identifier

1. dc.identifier=n78890351
2. dc.identifier=/bib.identifierAuthority=lccn n78890351

Notes:

- In the first example above, the identifier is an LCCN. This query could be used on a server where lccn is the default identifier type.

E.7.5 Examples of Searching by Date

1. **bib.dateIssued=2001** AND bib.namePersonal="matilda plews"
2. **bib.dateIssued=/dateAuthority=edtf 2001** AND bib.namePersonal="matilda plews"
3. **dc.date=2001**

Notes:

- Examples 1 and 2 have identical semantics since 'edtf' is the default date authority.
- To search simply on date where no qualification ("created", "published", etc.) is intended, dc.date should be used, as in example 3.

E.7.6 Examples of Searching by Format

1. **dc.format=/bib.formatAuthority=modsPhysicalForm print** AND bib.namePersonal="matilda plews"

Notes:

- modsPhysicalForm refers to the list at http://www.loc.gov/marc/sourcecode/form/formlist.html

E.7.7 Examples of Searching by Resource Type/Genre

1. **bib.genre=/bib.genreAuthority=modsGenre "humor, satire"** AND bib.namePersonal="dan jenkins"
2. **bib.genre=humor** AND bib.namePersonal="dan jenkins"
3. **dc.type=/bib.typeAuthority=modsResource text** AND bib.namePersonal="matilda plews"

Notes:

- bib.genre is for use with a controlled vocabulary. If the authority is omitted then a default is assumed (specified in the server's Explain information).
- 'modsGenre' refers to the list at http://www.loc.gov/marc/sourcecode/genre/genrelist.html.
- 'modsResource' refers to the enumerated list for resourceType in the MODS schema.
- Although as noted above, no bib index is defined for resource type and instead dc.type should be used, for bibliographic searching by genre, bib.genre, not dc.type, should be used (even though in general Dublin Core element type covers genre).

E.7.8 Examples of Searching by Target Audience

1. **bib.audience=/bib.audienceAuthority=modsAudience adolescent** AND bib.namePersonal="matilda plews"
2. **bib.audience=adolescent** AND bib.namePersonal="matilda plews"

Notes:

- This index is for use with a controlled vocabulary. If the authority is omitted then a default is assumed (specified in the server's Explain information).
- 'modsAudience' refers to the list at http://www.loc.gov/marc/sourcecode/target/targetlist.html.

E.7.9 Examples of Searching by Classification

1. bib.classification=RF110-320
2. bib.classification=/bib.classAuthority=lcc RF110-320

Notes:

- This index is for use with a controlled vocabulary. If the authority is omitted then a default is assumed (specified in the server's Explain information).
- 'lcc' as the class authority means the value is from the list at: http://www.loc.gov/marc/sourcecode/classification/classificationsource.html, and refers to "Library of Congress classification". For the example, click on "R" and then "Subclass RF" see that RF110-320 is the classification for "Otology. Diseases of the ear".

E.7.10 Examples of Searching by Place of Origin

1. bib.originPlace=london AND bib.namePersonal="jack t. ripper"

2. **bib.originPlace=/bib.geoUnit=country/bib.placeAuthority=marcCC cu** AND bib.namePersonal="livan hernandez"
3. **bib.originPlace=/bib.geoUnit=country/bib.placeAuthority=marcCN cuba** AND bib.namePersonal="livan hernandez"
4. **bib.originPlace=/bib.geoUnit=city havana** AND bib.namePersonal="livan hernandez"

Notes:

- http://www.loc.gov/marc/countries/ lists countries by name and code. marcCC is for country code and marcCN is for country name.

E.7.11 Examples of Searching by Language

- **dc.language=english** AND bib.subjectPlace=london
- **dc.language=/languageAuthority=iso639-2b car** AND bib.subjectPlace=carribean

E.7.12 Examples of Searching by Edition

- bib.edition=canadian

E.7.13 Examples of Searching by Part

- dc.title="neurology now" AND **bib.volume=1** AND **bib.issue=2**

E.7.14 Examples of Searching by Issuance

- dc.title="neurology now" AND bib.issuance=continuing

Appendix F. Query Type ‘cql-form’

Non-normative Annex

This Annex describes the query type ‘cql-form’.

The identifier (URI) for this query is **http://www.loc.gov/sru/oasis/cql-form**

The recommended short name to be used for the value of the parameter queryType in an SRU request is ‘**cql-form**’.

When the query type in an SRU query is ‘cql-form’ then the following parameters may occur in the SRU request:

- **Index Parameters**. Parameters with names of the form **qN.index,** where N is a positive integer. E.g. q1.index, q2.index, etc. The value of an index parameter is an index name.
- **Relation Parameters.** Parameters with names of the form **qN.relation,** where N is a positive integer. E.g. q1.relation, q2.relation, etc. The value of a relation parameter is a relation.
- **Term Parameters**. Parameters with names of the form **qN.term,** where N is a positive integer. E.g. q1.term, q2.term, etc. The value of a term parameter is a term.
- **Boolean Parameters**. Parameters with names of the form **qN.boolean,** where N is a positive integer. The value is a boolean, i.e. ‘’and’, ‘or’, or ‘and-not’. E.g. q1.and, q2.or.
- The parameter ‘boolean’, whose value is ‘pre’ or ‘post’.

The server processes the parameters as follows:

- For any given value of N, the server groups together all parameters whose names begin qN, and groups them into a search clause.
- For each clause, the term parameter must be present, and the index parameter is present if and only if the relation parameter is present; otherwise the query is in error.
- Call the search clause for integer N “clauseN” and the Boolean for integer N “booleanN”. For any N, if there is a booleanN there must be a clauseN, otherwise the query is in error. (There may be a clauseN with no booleanN.)
- Consider every clauseN and booleanN to be a token. If the value of the parameter ‘boolean’ is ‘pre’, then the server orders the tokens as:

  (first boolean) first clause (second boolean) second clause, etc.

  I.e. in increasing value of N for pairs of: booleanN, clauseN (or just clauseN if there is no booleanN); The result should be a valid prefix query; otherwise the query is in error.
- If the value of the parameter ‘boolean’ is ‘post’ then the boolean parameters follow rather than precede the clause. The result should be a valid postfix query; otherwise the query is in error.
- If the result is a valid prefix or postfix query, the server proceeds to process it.