

The Library of Congress » Librarians, Archivists » Standards

SRU Pages

SEARCH

SRU Version 1.1 13th February 2004

SRU: SEARCH / RETRIEVE VIA URL

[Home](#) > > [Search/Retrieve](#)

SRU SEARCH/RETRIEVE OPERATION

[searchRetrieve Request](#) - [request parameters](#) - [searchRetrieve Response](#) - [response parameters](#)
[SRU Syntax](#) - [Encoding](#) - [Result Set Model](#) - [Record Retrieval](#) - [sorting](#) - [Diagnostics](#)
[version information](#) - [CQL](#) - [Examples](#)

SRU SearchRetrieve Request

Syntax

An SRU request is a URI as described in [RFC 3986](#). Specifically it is an HTTP URL (as described in section 3.3 of [RFC 1738](#); however see [note 1](#)). It consists of an SRU base url and a searchpart (see [note 2](#), separated by a question mark ("?").

For example, in the SRU request:

<http://z3950.loc.gov:7090/voyager?version=1.1&operation=searchRetrieve&query=dinosaur>

the base URL is "http://z3950.loc.gov:7090/voyager" and the searchpart is "version=1.1&operation=searchRetrieve&query=dinosaur".

The SRU request may include Unicode characters (characters from the Universal Character Set, ISO 10646). See [Encoding](#).

The searchpart consists of parameters separated by "&", each with structure "key=value". The names of the parameters from the [table below](#) are the "key" strings within the URL.

If there are no parameters (an [Explain](#) request) the SRU request may consist of the base-url with or without the trailing '?'. When there is one parameter or more, the URL may end with an '&' or omit the trailing '&' (the end of a URI query string implicitly terminates the last parameter value).

Notes:

- 1. RFC 1738 is obsoleted by RFC 3986. However, RFC 1738 describes the 'http:' URI scheme; RFC 3986 does not, instead indicating that a separate document will be written to do so, but it has not yet been written. So currently there is no valid, normative reference for the 'http:' URI scheme, and so the obsolete RFC 1738 is referenced. When there is a valid, normative reference, it will be listed here.*
- 2. The searchpart is known as the "query" in URI terminology. It is called the searchpart here to avoid confusion with the CQL query (which is one of the parameters of the*

searchpart).

Encoding (and Support for Unicode Characters)

the following encoding procedure is recommended, in particular, to accommodate Unicode characters (characters from the Universal Character Set, ISO 10646) beyond U+007F, which are not valid in a URI.

1. Convert each key name and value to UTF-8.
2. Percent-encode characters as necessary within each key name and value. See <http://rfc.net/rfc3986.html#s2.1>.
3. Construct a searchpart (a URI query) from the encoded key names and values, as described in [Syntax](#). (For each key name/value pair, the key name and value are combined with '='; the resulting key name/value combinations are combined by '&'.)

Note: In step 2, it is recommended to percent-encode every character in a key name or value that is not in the URI unreserved set, that is, all except alphabetic characters, decimal digits, and the following four special characters: dash(-), period (.), underscore (_), tilde (~). (By this procedure some characters may be percent-encoded that do not need to be -- For example '?' occurring in a value does not need to be percent encoded, but it is safe to do so. If in doubt, percent-encode.)

Example

Consider the following parameter:

```
query=dc.title=/x kirkegård
```

The key name is "query" and the value is "dc.title=/x kirkegård"

Note that the first '=' (following "query") *must not* be percent encoded as it is used as a URI delimiter, it is not part of a key name or value. The second '=' (preceding the '/') *must* be percent encoded as it is part of a key value.

The following characters must be percent encoded:

- ▶ the second '=', percent encoded as %3D
- ▶ the '/', percent encoded as %2F
- ▶ the space, percent encoded as %20
- ▶ the 'å'. Its UTF-8 representation is C3A5, two octets, and correspondingly it is represented in a URI as two characters percent encoded as %C3%A5.

Server Procedure

1. Parse received request based on '?', '&', and '=' into component parts: the base URL, and parameter names and values from key names and values.
2. For each component:

- a. Decode all %-escapes.
- b. Treat the result as a UTF-8 string.

SRU SearchRetrieve Request Parameters

Parameter Name	Mandatory or optional	Description
version	mandatory	The version of the request, and a statement by the client that it wants the response to be less than, or preferably equal to, that version.
query	mandatory	Contains a query expressed in CQL to be processed by the server. <i>Note: support for CQL is required in order for a client or server to claim conformance to the SRU protocol.</i>
startRecord	optional	The position within the sequence of matched records (see result set model) of the first record to be returned. The first position in the sequence is 1. The value supplied must be greater than 0. Default value if not supplied is 1.
maximumRecords	optional	The number of records requested to be returned. The value must be 0 or greater. Default value if not supplied is determined by the server (see explain). The server may return less than this number of records, for example if there are fewer matching records than requested. See additional description .
recordPacking	optional	A string to determine how the record should be escaped in the response. Defined values are 'string' and 'xml'. The default is 'xml'. See additional description .
recordSchema	optional	The schema requested for the records to be returned.

		<p>If the recordXPath parameter (next) is included, it is the abstract schema for purposes of evaluation by the XPath expression.</p> <p>If the recordXPath parameter is not included, it is the schema that response records should assume.</p> <p>The value is the URI identifier for the schema or the short name for it published by the server. The default value if not supplied is determined by the server (see explain). See additional description.</p>
recordXPath	optional	An XPath expression, to be applied to the records before returning them. It is to be applied relative to the schema supplied in the recordSchema parameter, and response records should assume the SRU XPath schema .
resultSetTTL	optional	The number of seconds for which the client requests that the result set created should be maintained. The server may choose not to fulfil this request, and may respond with a different number of seconds (parameter resultSetIdleTime in response). If resultSetTTL is not supplied then the server will determine the value.
sortKeys	optional	Contains a sequence of sort keys to be applied to the results. See Sort .
stylesheet	optional	A URL for an xml stylesheet. The client requests that the server simply return this URL in the response. See additional description .
extraRequestData	optional	Provides additional, profile specific information. See Extra Data .
operation	mandatory	The string: 'searchRetrieve'.

Example:

<http://z3950.loc.gov:7090/voyager?version=1.1&operation=searchRetrieve&query=dinosaur&maximumRecords=1&recordSchema=dc>

This example is a search for the term "dinosaur", requesting that at most one record be returned, according to the 'dc' schema.

SRU searchRetrieve Response

The response to an SRU searchRetrieve request is an XML document (for its schema, see searchRetrieveResponseType defined within [srw-types.xsd](#)). The table below provides a summary and description of the elements provided by the XML document. The "Type" column indicates either an XML type ("xsd:") or a type defined within the schema.

SRU SearchRetrieve Response Parameters

Name	Type	Mandatory or Optional	Description
version	<i>xsd:string</i>	Mandatory	The version of the response. Must be less than or equal to the version requested by the client.
numberOfRecords	<i>xsd:integer</i>	Mandatory	The number of records matched by the query. If the query fails this will be 0.
resultSetId	<i>xsd:string</i>	Optional	The identifier for a result set that was created through the execution of the query. See additional description . See also result set model .
resultSetIdleTime	<i>xsd:integer</i>	Optional	The number of seconds after which the created result set will be deleted. (Not guaranteed. The result set may become unavailable before this.) See additional

			description . See also result set model .
records	<i>sequence of <record></i>	Optional	A sequence of records matched by the query, or surrogate diagnostics. See additional description below.
nextRecordPosition	<i>xsd:integer</i>	Optional	The next position within the result set following the final returned record. If there are no remaining records, this field should be omitted.
diagnostics	<i>sequence of <diagnostic></i>	Optional	A sequence of non surrogate diagnostics generated during execution.
extraResponseData	<i><xmlFragment></i>	Optional	Additional, profile specific information See Extra Data .
echoedSearch RetrieveRequest	<i><echoedSearch RetrieveRequest></i>	Optional	The request parameters echoed back to the client in a simple XML form.

Result Set Model

SRU does not assume support of persistent result sets -- that a result set created by one request may necessarily be accessed by a client in a subsequent request. SRU does expect the server to state whether or not it supports persistent result sets, and if so the result set model described [below](#) is assumed.

There are applications in which result sets are critical; on the other hand there are applications in which result sets are not viable. An example of the first might be scientific investigation of a database with comparison of data sets produced at different times. An example of the latter might be a very frequently used database of web pages in which persistent result sets would be an impossible burden on the infrastructure due to the frequency of use.

Even if the server does not make result sets available for public manipulation, the following

model is also important to understand in order to allow a single request to both match records and then sort them.

Model

Processing of a query results in the selection of a set of records, represented by a result set maintained at the server; logically it is an ordered list of references to the records. Once created, a result set cannot be modified. Any operation which would somehow change a result set instead creates a new result set. Each result set is referenced via a unique identifying string, generated by the server when the result set is created.

From the client's point of view, the result set is a set of records each referenced by an ordinal number, beginning at 1. The client may request a given record from a result set according to a specific schema. For example the client may request record 1 in Dublin Core, and subsequently request record 1 in MODS. The requested schema is not a property of the result set (nor of the requested records as a member of the result set); the result set is simply the ordered list of records.

A record might be deleted or otherwise become unavailable while a result sets which references that record still exists. If a client then requests that record, the server is expected to supply a surrogate diagnostic in place of the record. For example, if the record at position 2 in a result set is deleted and then a client requests records 1 through 3, the server should supply, in order: record 1, a surrogate diagnostic for record 2, record 3.

The records in a result set are not necessarily ordered according to any specific or predictable scheme, unless it has been created with a request that contains one or more sort keys. See [sort](#) for more information regarding the specifics of sorting. If search and sort specifications are supplied on the same request then only the final sorted result set is considered to exist, even if the server internally creates a result set and then sorts it.

Record Retrieval

All records in SRU are transferred in XML. (Records are not assumed to be stored in XML. Records which are not natively XML must be first transformed into XML before transfer.) Records in the response may be expressed as a single string, or as embedded XML. If a record is transferred as embedded XML, it must be well formed and should be validatable against the record schema (see recordSchema parameter below).

The records parameter in the response is a sequence of 'record' elements, each of which contains either a record or a surrogate [diagnostic](#) explaining why that particular record could not be transferred. Each 'record' element is structured into the following elements.

Name	Type	Mandatory or Optional	Description
------	------	-----------------------	-------------

recordSchema	<i>xsd:string</i>	mandatory	The URI identifier of the XML schema in which the record is encoded. Although the request may use the server's assigned short name, the response must always be the full URI.
recordPacking	<i>xsd:string</i>	mandatory	The packing used in recordData, as requested by the client or the default. Additional description.
recordData	<i><stringOrXmlFragment></i>	mandatory	The record itself, either as a string or embedded XML
recordPosition	<i>xsd:positiveInteger</i>	optional	The position of the record within the result set. (Recommended for use in SRU, may be returned in SRW as well.)
extraRecordData	<i><xmlFragment></i>	optional	Any additional information to be transferred with the record. See extra data .

An example record, in the simple Dublin Core schema, packed as XML:

```

<record>
  < recordSchema>info:srw/schema/1/dc-v1.1</recordSchema>
  < recordPacking>xml</recordPacking>
  < recordData>
    < srw_dc:dc>

```

```

    <dc:title>This is a Sample Record</dc:title> </srw_dc:dc>
  < /recordData>
  <recordPosition>1</recordPosition>
  <extraRecordData>
    <rel:rank>0.965</rel:rank>
  < /extraRecordData>
< /record>

```

Sorting

A request may include a sort specification, indicating the desired ordering of the results. This is a request for the server to apply a sorting algorithm to the result set before returning any records. It may be supplied with a new search or applied to an existing result set.

The sort parameter is included in the searchRetrieve operation, rather than defined as a separate operation, for two reasons: (1) if the server knows the desired sort order in advance, query processing can be optimized; (2) a server may be able to sort a result set at creation, but not maintain it across multiple requests.

In order to specify result set(s) to sort, the query should include:

```
cql.resultSetId = "resultSetId"
```

where 'resultSetId' is the identifier supplied by the server for the result set. If multiple result set identifiers are supplied, linked by boolean OR, AND or NOT, then the request will combine and sort all of the given sets together. This is documented in the [CQL context set](#).

The sort parameter includes one or more keys, each of which includes the following information:

Name	Type	Required	Description
path	<i>xsd:string</i>	Mandatory	An XPath expression describing a tagpath to be used in the sort. See additional description .
schema	<i>xsd:string</i>	Optional	The URI identifier for a supported schema. This schema is the one to which the XPath expression applies. If it is not supplied then the default value from Explain will be used. See additional description .
ascending	<i>xsd:boolean</i>	Optional	Should the results be sorted ascending (true, and the default if not supplied) or

			descending (false).
caseSensitive	<i>xsd:boolean</i>	Optional	Should case be considered as important during the sort. The default value is false if not supplied.
missingValue	<i>xsd:string</i>	Optional	One of 'abort', 'highValue', 'lowValue', 'omit' or a supplied value. The semantics of each are described below and the default is 'highValue'.

XPath and Schema

[XPath](#) is a W3C specification which allows the description of an element path. So to sort by title, one might specify the xpath of "/record/title" within the Dublin Core schema. The records need not be stored in this particular schema to be able to sort by it. The records do not even necessarily need to be able to be returned in the schema.

SRU has the concept of utility schemas which are designed not to return records in, but into which records can be transformed in order to sort them in a particular way. For example, if the record has a geographical location in it, then it may be desirable to sort the locations in the records from north to south and east to west. This would obviously require transformation into a schema that allows sorting by a convenient coordinate system, rather than lexically on the place name, and this schema may not be available for retrieving the records.

Missing Value Action

This parameter of a sort key instructs the server what to do when the supplied XPath is not present within the record. For example if the server is instructed to sort by author, and a record has no author, it will behave in accordance with this value.

Its value may be:

- ▶ **abort**: The server should immediately cease trying to sort and return a diagnostic that the sort could not be performed.
- ▶ **omit**: The server should remove this record from the results.
- ▶ **lowValue**: The server should sort this as if it were the lowest possible value.
- ▶ **highValue**: The server should sort this as if it were the highest possible value.
- ▶ **"constant"**: The server should sort the record as if this value ("constant") were supplied.

sortKeys

The textual representation of a sort key is achieved by the following rules.

1. The path must be included as the first parameter.
2. Subsequent parameters are separated by the use of a comma (,) character in the order given above.
3. The path and schema must be quoted if they contain quotes, commas or spaces. Internal quotes must be escaped with a backslash.
4. Parameters beyond the first may be supplied with no value, in which case the server will use the default.
5. The last parameter supplied must be present. (In other words, the key may not end in a comma.)
6. Boolean parameters are expressed as 1 (true) or 0 (false).
7. Multiple keys are separated by whitespace.

An example of the sortKeys parameter in an SRU URL might be:

```
&sortKeys=title,onix date,onix,0
```

This example asks to sort primarily by 'title', ascending (as the default), from the 'onix' schema; and secondarily by 'date', descending, also from the 'onix' schema. 'caseSensitive' and 'missingValue' are not given, therefore the defaults apply, namely 'insensitive' and 'highValue'.

And a complete SRW sortKeys parameter might be:

```
<sortKeys>  
"/record/title","info:srw/schema/1/dc-v1.1",1  
"/record/datafield[@tag="100"]/subfield[@code="a"]",  
"http://www.loc.gov/MARC21/slim/","Smith"  
</sortKeys>
```

Failure to Sort

If the server is unable to create a sorted result set according to the request, then it must supply appropriate diagnostics stating this. See the [diagnostics](#) specification for more information.

xSortKeys

The XML representation of sort keys is very simple. Each key is wrapped in a 'sortKey' element. The elements within are the parameters described above. The path element is required, but the others are optional. This is used only when echoing the searchRetrieveRequest for SRU.

An example of xSortKeys:

```
<xSortKeys> <sortKey> <path>/record/title</path>
<schema>info:srw/schema/1/dc-v1.1</schema> <ascending>true</ascending>
</sortKey> <sortKey>
<path>/record/datafield[@tag="100"]/subfield[@code="a"]</path>
<schema>http://www.loc.gov/MARC21/slim/</schema>
<missingValue>"Smith"</missingValue> </sortKey> </xSortKeys>
```

Additional Parameter Description

Record Packing

In order that records which are not well formed do not break the entire message, it is possible to request that they be transferred as a single string with the < > and & escaped to their entity forms. Moreover some toolkits may not be able to distinguish record XML from the XML which forms the response. However, some clients may prefer that the records be transferred as XML in order to manipulate them directly with a stylesheet which renders the records and potentially also the user interface.

This distinction is made via the recordPacking parameter in the request. If the value of the parameter is 'string', then the server should escape the record before transferring it. If the value is 'xml', then it should embed the XML directly into the response. Either way, the data is transferred within the 'recordData' field. If the server cannot comply with this packing request, then it must return a [diagnostic](#).

maximumRecords

A query may be either a search, a reference to an existing result set or a combination of searches and resultsets. In any case, records may be transferred from a successful query if the maximumRecords parameter was either positive or not supplied. If the maximumRecords parameter is 0, then no records should be transferred. If it is not supplied then the server is at liberty to transfer any number of records it desires. The number of records transferred may be less than the value of maximumRecords, but may not be greater than it. For example if the end of the result set is reached, then fewer records than requested may be transferred. The maximum number of records that a server will transfer in one transaction is listed in the Explain setting 'maximumRecords'.

recordSchema

This parameter identifies the schema assumed for records to be returned. Its value is either

a short name for a record schema, or the full URI which identifies it. The short name, if supplied, must be one advertised in the explain record for the server. For example, if the explain record advertises that the Dublin Core schema's short name is 'dc', then the client may send either 'dc' or 'info:srw/schema/1/dc-v1.1' and expect to receive the record in unqualified Dublin Core. See [SRU Record Schemas](#).

(If the recordXPath parameter is included, this parameter identifies the abstract schema for purposes of evaluation by the XPath expression. If the recordXPath parameter is not included, it is the schema that response records should assume.)

If the recordSchema parameter is included in the request, then every record returned should be according to the requested schema. If the server cannot return a particular record according to that schema it should substitute a surrogate diagnostic (info:srw/diagnostic/1/67). If the server does not recognize/support the schema, it should fail the request, supplying a non-surrogate diagnostic (e.g. info:srw/diagnostic/1/66 - "schema unknown")

If the recordSchema parameter is omitted in the request, the server may return records in any schema, and may return different records in different schemas.

resultSetId

If the server supports result sets, it may include a resultSetId in the searchRetrieve response, along with an idle time described below. If another query is submitted then the server will again supply a result set id. If the result of the query would modify an existing result set (for example, a request to sort an existing result set), then the server must supply a new id for this new set. The server should maintain unique names for each result set created, even if the result sets no longer exist, such that clients do not mistakenly request records from the new set when meaning to refer to the previous set with the same identifier.

resultSetIdleTime

The server may supply an idle time along with a result set. The server is making a good-faith estimate that the result set will remain available and unchanged (both in content and order) until a timeout (a period of inactivity exceeding the idle time). The idle time is an integer representing seconds; it must be a positive integer, and should not be so small that a client cannot realistically reference the result set again. If the server does not intend that the result set be referenced, it should omit the result set identifier in the response.

stylesheet

In order to render the response, "thin" clients may provide a stylesheet to turn the response XML into a natively renderable format, often HTML or XHTML. This allows a web browser,

or other application capable of rendering stylesheets, to act as a dedicated client without requiring any further application logic. The echoedRequest parameter on the response enables a client to use this stylesheet to also have the request it just made available without any client side logic.

The URL to be included in the response is given in the request in the 'stylesheet' parameter. This URL is to be included in the href attribute of the xml-stylesheet processing instruction before the response. It is likely that the type will be XSL, but not necessarily, and this may be changed by profiles. If the server cannot fulfill this request it must supply a [diagnostic](#).

This parameter may only be used with SRU. It is a SOAP error to return a stylesheet in SRW, and hence an error to request one.

If this parameter is not supplied, then the server can, at its discretion, include a default stylesheet. The default stylesheet URL may be included in the explain document.

For example, if the client includes 'stylesheet=/master.xml', then the server must include at the very beginning of the response:

```
<?xml-stylesheet type="text/xsl" href="/master.xml"?>
```

SRU Examples

A typical SRU request is an HTTP GET URL looking like this:

```
http://myserver.com/myurl/?operation=searchRetrieve
& version=1.1&query=dc.identifier+%3d%220-8212-1623-6%22
&recordSchema=dc&recordPacking=XML
& stylesheet=http://myserver.com/myStyle
& x-info-2-auth1.0-authenticationToken=%22XDFPQR5ZZ%22
```

And the response is an XML document:

```
<?xml-stylesheet type="text/xsl"
  href="http://myserver.com/myStyle"?>
<srw:searchRetrieveResponse
  xmlns:srw="http://www.loc.gov/zing/srw/"
  xmlns:diag="http://www.loc.gov/zing/srw/diagnostic/"
  xmlns:xcql="http://www.loc.gov/zing/cql/xcql/"
  xmlns:dc="http://purl.org/dc/elements/1.1/">
  <srw:version>1.1</srw:version>
  <srw:numberOfRecords>1</srw:numberOfRecords>
  <srw:resultSetId>
    8c527d60-c3b4-4cec-a1de-1ff80a5932df
  </srw:resultSetId>
  <srw:resultSetIdleTime>600</srw:resultSetIdleTime>
  <srw:records>
    <srw:record>
```

```
<srw:recordPacking>XML</srw:recordPacking>
<srw:recordSchema>
  info:srw/schema/1/dc-v1.1
</srw:recordSchema>
<srw:recordData>
  <dc:dc>
    <dc:title>Robert Frank</dc:title>
    <dc:creator>Robert Frank 1924-</dc:creator>
    <dc:publisher>Houston</dc:publisher>
    <dc:date>1986</dc:date>
    <dc:identifier>0-8212-1623-6</dc:identifier>
  </dc:dc>
</srw:recordData>
<srw:recordNumber>1</srw:recordNumber>
</srw:record>
<srw:record>
  <srw:recordPacking>XML</srw:recordPacking>
  <srw:recordSchema>
    info:srw/schema/1/diagnostic-v1.1
  </srw:recordSchema>
  <srw:recordData>
    <diag:diagnostic>
      <diag:uri>info:srw/diagnostic/1/71</diag:uri>
      <diag:detail>XML</diag:detail>
      <diag:message>
        Unsupported record packing
      </diag:message>
    </diag:diagnostic>
  </srw:recordData>
  <srw:recordNumber>2</srw:recordNumber>
</srw:record>
</srw:records>
<srw:echoedSearchRetrieveRequest>
  <srw:version>1.1</srw:version>
  <srw:query>dc.identifier="0-8212-1623-6"</srw:query>
  <srw:xQuery>
    <xcql:searchClause>
      <xcql:index>dc.identifier</xcql:index>
      <xcql:relation>
        <xcql:value>=</xcql:value>
      </xcql:relation>
      <xcql:term>0-8212-1623-6</xcql:term>
    </xcql:searchClause>
  </srw:xQuery>
  <srw:recordSchema>dc</srw:recordSchema>
  <srw:recordPacking>XML</srw:recordPacking>
  <srw:stylesheet>
    http://myserver.com/myStyle
  </srw:stylesheet>
  <srw:extraResponseData>
    <rob:authenticationToken
      xmlns:rob="info:srw/extension/2/auth-1.0">
      XDFPQR5ZZ
```

```
</rob:authenticationToken>  
</srw:extraResponseData>  
</srw:echoedSearchRetrieveRequest>  
</srw:searchRetrieveResponse>
```

[Home](#) >> **Search/Retrieve**

The Library of Congress » Librarians, Archivists » Standards
December 7, 2005

[Contact Us](#)