searchRetrieve: Part 6. SRU Scan Operation Version 1.0

Committee Specification Draft 01 / Public Review Draft 01

08 December 2011

Specification URIs

This version:

http://docs.oasis-open.org/search-ws/searchRetrieve/v1.0/csprd01/part6-scan/searchRetrieve-v1.0-csprd01-part6-scan.doc (Authoritative)
http://docs.oasis-open.org/search-ws/searchRetrieve/v1.0/csprd01/part6-scan/searchRetrieve-v1.0-csprd01-part6-scan.html
http://docs.oasis-open.org/search-ws/searchRetrieve/v1.0/csprd01/part6-scan/searchRetrieve-v1.0-csprd01-part6-scan.pdf

Previous version:

N/A

Latest version:

http://docs.oasis-open.org/search-ws/searchRetrieve/v1.0/searchRetrieve-v1.0-part6-scan.doc (Authoritative)
http://docs.oasis-open.org/search-ws/searchRetrieve/v1.0/searchRetrieve-v1.0-part6-scan.html
http://docs.oasis-open.org/search-ws/searchRetrieve/v1.0/searchRetrieve-v1.0-part6-scan.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/csprd01/schemas/
- *searchRetrieve: Part 0. Overview Version 1.0*. http://docs.oasis-open.org/search-ws/searchRetrieve/v1.0/csprd01/part0-overview/searchRetrieve-v1.0-csprd01-part0-overview.html
- *searchRetrieve: Part 1. Abstract Protocol Definition Version 1.0*. http://docs.oasis-open.org/search-ws/searchRetrieve/v1.0/csprd01/part1-apd/searchRetrieve-v1.0-csprd01-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/csprd01/part2-sru1.2/searchRetrieve-v1.0-csprd01-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/csprd01/part3-sru2.0/searchRetrieve-v1.0-csprd01-part3-sru2.0.html
- *searchRetrieve: Part 4. APD Binding for OpenSearch Version 1.0*. http://docs.oasis-open.org/search-ws/searchRetrieve/v1.0/csprd01/part4-opensearch/searchRetrieve-v1.0-csprd01-part4-opensearch.html
- *searchRetrieve: Part 5. CQL: The Contextual Query Language Version 1.0*. http://docs.oasis-open.org/search-ws/searchRetrieve/v1.0/csprd01/part5-cql/searchRetrieve-v1.0-csprd01-part5-cql.html
- *searchRetrieve: Part 6. SRU Scan Operation Version 1.0*. (this document) http://docs.oasis-open.org/search-ws/searchRetrieve/v1.0/csprd01/part6-scan/searchRetrieve-v1.0-csprd01-part6-scan.html
- *searchRetrieve: Part 7. SRU Explain Operation Version 1.0*. http://docs.oasis-open.org/search-ws/searchRetrieve/v1.0/csprd01/part7-explain/searchRetrieve-v1.0-csprd01-part7-explain.html

Related work:

- Scan Operation. Library of Congress. http://www.loc.gov/standards/sru/specs/scan.html

Abstract:

This is one of a set of documents for the OASIS Search Web Services (SWS) initiative. This document, "*SRU Scan Operation"* is the specification of the scan protocol. Scan is a companion protocol to the SRU protocol which enables searches for specific terms; scan allows the client to request available terms that may be searched.

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:

[SearchRetrievePt6]

searchRetrieve: Part 6. SRU Scan Operation Version 1.0. 08 December 2011. OASIS Committee Specification Draft 01 / Public Review Draft 01. http://docs.oasis-open.org/search-ws/searchRetrieve/v1.0/csprd01/part6-scan/searchRetrieve-v1.0-csprd01-part6-scan.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 the specification of the Explain Operation.

The documents in this collection of specifications are:

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

Scan is a companion protocol to SRU1.2 and SRU2.0 (the Search and Retrieve via URL protocol).

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 Overview and Model

While the searchRetrieve operation enables searches for specific terms within the records, the scan operation allows the client to request a range of the available terms at a given point within a list of indexed terms. This enables clients to present an ordered list of values and (if supported) how many hits there would be for a search on a given term. Scan is often used to select terms for subsequent searching or to verify a negative search result.

2.1 Operation Model

The SWS initiative defines three operations:

1. **SearchRetrieve Operation**. The main operation. The SRU protocol defines a request message (sent from an SRU client to an SRU server) and a response message (sent from the server to the client). This transmission of an SRU request followed by an SRU response constitutes a SearchRetrieve operation.
2. **Scan Operation**. The Scan operation is defined by the Scan protocol, which is this specification. Similar to SRU, it defines a request message and a response message. The transmission of a Scan request followed by a Scan response constitutes a Scan operation.
3. **Explain Operation**. See Explain Model. When a client retrieves an Explain record, this constitutes an Explain operation.

Note: In earlier versions a searchRetrieve or scan request carried a mandatory operation parameter. In version 2.0, there is no operation parameter for either. See Interoperation with Earlier Versions.

2.2 Data model

Search engines often create indexes on the fields that they search. These indexes can consist of all or part of the contents of single fields or combinations of fields from records in their database. Some of these indexing search engines are capable of exposing the lists of search terms that they have generated. An exposable list of search terms is called a **scanable index** (or **index** when it is clear from the context that “scanable index” is meant.)

Each scanable index is sorted according to an order that is defined by the server and may be different for different indexes.

2.3 Protocol Model

The protocol model assumes these conceptual components:

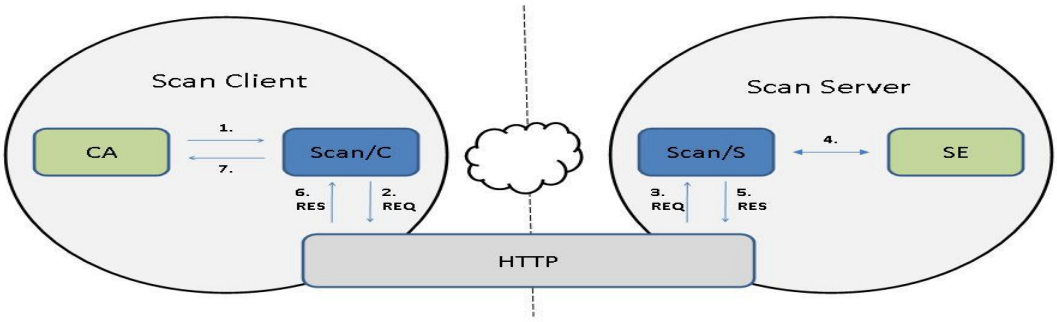
- The client application (**CA**),
- the Scan protocol module at the client (**Scan/C**),
- the lower level protocol (**HTTP**),
- the Scan protocol module at the server (**Scan/S**),
- the search engine at the server (S**E**).

For modeling purposes this standard assumes but does not prescribe bindings between the CA and Scan/C and between Scan/S and SE, as well as between Scan/C and HTTP and between Scan/S and HTTP; for examples of the latter two see Bindings to Lower Level Protocols. The conceptual model of protocol interactions is as follows:

- At the client system the Scan/C accepts a request from the CA, formulates a searchRetrieve protocol request (**REQ**) and passes it to HTTP.
- Subsequently at the server system HTTP passes the request to the Scan/S which interacts with the SE, forms a searchRetrieve protocol response (**RES**), and passes it to the HTTP.

- At the client system, HTTP passes the response to the Scan/C which presents results to the CA.

The protocol model is described diagrammatically in the following picture:



1. CA passes a request to Scan/C.
2. Scan/C formulates a REQ and passes it to HTTP.
3. HTTP passes the REQ to Scan/S.
4. Scan/S interacts with SE to form a RES.
5. The RES is passed to HTTP.
6. HTTP passes the RES to Scan/C.
7. Scan/C presents results to CA.

2.4 Processing Model

The client provides the name of a scanable index, and a term that may or may not be in the index. The server locates either that term within the index or the term that is closest (in terms of the order defined for that index), and responds with an ordered list of terms, some before and/or some following the supplied term. The supplied term itself may or may not be in the index, and if not does not appear in the supplied list. (The numbers of terms preceding and/or following the supplied term are determined by parameters supplied in the request.)

2.5 Query model

Scan requires support for part of the CQL query language. Specifically, the scanClause which is part of the scan request takes the form of a CQL search clause. The following is supplied as a very cursory overview of CQL.

A CQL query consists of a single search clause, or multiple search clauses connected by Boolean operators: AND, OR, or AND-NOT. A search clause may include an index, relation, and search term (or a search term alone where there are rules to infer the index and relation). Thus for example “title = dog” is a search clause in which “title” is the index, “=” is the relation, and “dog” is the search term. “Title = dog AND subject = cat” is a query consisting of two search clauses linked by a Boolean operator AND, as is “dog AND cat”. CQL also supports proximity and sorting. For example, “cat prox/unit=paragraph hat” is a query for records with “cat” and “hat” occurring in the same paragraph. “title = cat sortby author” requests that the results of the query be sorted by author.

2.6 Diagnostic Model

Diagnostics can be returned for a number of reasons. Typically, these are fatal errors and no terms will be returned along with the diagnostic.

2.7 Explain Model

Every Scan server provides an associated Explain record, retrievable as the response of an HTTP GET at the base URL for the server. A Scan client may retrieve this record which provides information about the server's capabilities. The client may use the information in the Explain record to self-configure and provide an appropriate interface to the user.

The server lists the names of all indexes in its Explain file. For those indexes that are scanable, the attribute "scan" will be set to "true" in the <index> element of the index. (The absence of "scan='true'" on the <index> element does not necessarily mean that scan is not supported for that server.) The Explain file may also include sample requests, and conditions of use (for example mandatory display of copyright and syndication rights).

2.8 Serialization Model

Requests can be sent as HTTP GET requests. Some servers support POST requests with the parameters encoded as form elements. Responses are only defined for XML, but other response serializations, such as JSON are possible through use of either the httpAccept parameter or through content negotiation (when supported).

3 Scan Request

3.1 Summary of Request Parameters

The request parameters are summarized in the following table.

Table 1. Summary of Request Parameters.

Name	Occurrence	Description or Reference
scanClause	mandatory	See **scanClause**
responsePosition	optional	See **responsePosition and maximumTerms**
maximumTerms	optional	
httpAccept	optional	See httpAccept
stylesheet	optional	See stylesheet
extraRequestData	optional	See Extension Request Parameter

3.2 Request Parameter Descriptions

3.2.1 scanClause

The client supplies the parameter **scanClause** in the request, indicating the index to be scanned and the start point within the index.

The scanClause is expressed as a complete CQL search clause: index, relation, term. The term is the position within the ordered list of terms at which to start, and is referred to as the **start term**.

For example, the scanClause “title==cat” indicates the index ‘title’ and start term ‘cat’.

The relation and relation modifiers may be used to determine the format of the terms returned. For example 'title any cat' will return a list of keywords, whereas 'title == cat' would return a list of full title fields. Range relations such as ‘<’, ‘>’, ‘within’ may not be used.

3.2.2 responsePosition and maximumTerms

The client supplies the parameter **responsePosition** in the request, indicating the position within the list of terms returned where the client would like the start term to occur. Its value is an integer. The default value is server defined.

Note that the startTerm may or may not be part of the index. The expression **nearest term** means the startTerm if it is part of the index, or if it is not, the term nearest (as defined by the server) to where the startTerm would have been, if it had been part of the index.

The client also supplies the parameter **maximumTerms**, the number of terms which the client requests be supplied in the response. Its value is a positive integer and its default value if not supplied is determined by the server.

Let P and M be the value of responsePosition and maximumTerms respectively.

The first term in the list is determined as follows.

- If P is zero or less, the nearest term is not included. The first term in the list is the term that comes Q terms after the nearest term, where Q= |P|+1. (Absolute value of P plus 1) E.g., if P=-1, then the first term in the list should be the second term following the nearest term.

- If P is positive, the first term in the list should be the term that comes Q terms before the nearest term, where Q= P-1. (E.g., if P=3, this means that the nearest term should be third in the list which means that the first term in the list should be the second term preceding the nearest term.)
    - Note that if P exceeds M, then the start term is not included in the list; all members of the list precede the start term.

The actual number of terms supplied in the list SHOULD NOT exceed M, but may be fewer, for example if the end of the term list is reached.

Example

Suppose

- the index consists of the following terms in this order: A,B,C,D,E,F,G,H
- nearest term is D
- maximumTerms = 3

Then:

- If startTerm= -1, The list supplied will be F,G,H
- If startTerm= 0, the list supplied will be E,F,G
- If startTerm= 1, the list supplied will be D,E,F
- If startTerm=4, the list supplied will be A,B,C

3.2.3 httpAccept

The request parameter **httpAccept** may be supplied to indicate the preferred format of the response. The value is an internet media type. For example if the client wants the response to be supplied in the ATOM format, the value of the parameter is 'application/atom+xml'.

The default value for the response type is 'application/sru+xml'.

Note:This media type is pending registration. The pre-registration media type application/x-sru+xml should be accepted.

The intent of the httpAccept parameter can be accomplished with an HTTP Accept header. Servers SHOULD support either mechanism. In either case (via the httpAccept parameter or HTTP Accept header), if the server does not support the requested media type then the server MUST respond with a 406 status code and SHOULD return an HTML message with pointers to that resource in supported media types.

If a server supports multiple media types and uses content negotiation to determine the media type of the response, then the server SHOULD provide a URL in the Content-Location header of the response pointing directly to the response in that mime-type.

For instance, if the client had sent the URL

http://example.org/scan?scanClause=title=dog

with an Accept header of 'application/rss+xml',

then the server SHOULD return a Content-Location value of http://example.org/sru?scanClause=title=dog&httpAccept=application/rss+xml. This Content-Location header is returned along with the content itself, presumably in the application/rss+xml format. (It would also be acceptable to return a redirect to that URL instead, but that behavior is not encouraged as it is inefficient.)

The default response type is application/sru+xml. That is, if there is neither an Accept header (or if there is an Accept header of "*") nor an httpAccept parameter, the response should be of media type

application/sru+xml, and a corresponding Content-Location header should be returned with the response. For example if the request is

http://example.org/scan?query=dog

a Content-Location header of

http://example.org/scan?query=dog&httpAccept=application/sru+xml

should be returned.

3.2.4 Stylesheet

The request parameter '**stylesheet**' is a URL for a stylesheet, to be used for the display of the response to the user. The client requests that the server simply return this URL in the response, in the href attribute of the xml-stylesheet processing instruction before the response xml. (It is likely that the type will be XSL, but not necessarily so.) If the server cannot fulfill this request it MUST supply a non-surrogate diagnostic .

The purpose is to allow a thin client 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.

```
Example
http://z3950.loc.gov:7090/voyager?stylesheet=/master.xsl&query=dinosaur
```

This requests the server to include the following as beginning of the response:

```
<?xml version="1.0"?>
 <?xml-stylesheet type="text/xsl" href="/master.xsl"?>
 <scan:scanResponse ...
```

3.3 Serialization of Request Parameters

- **Example:** *Get 25 title terms centered on the word "frog"*
    - http://myserver.com/scan?scanClause=dc.title = frog &responsePosition=13&maximumTerms=25
- **Example:** *If the last term returned by the first example was "goat", ask for the term "goat" and the 24 terms that follow it. From a user perspective, this would be a "page down" through the ordered list of title terms*
    - http://myserver.com/scan?scanClause=dc.title = goat &responsePosition=1&maximumTerms=25
- **Example**: *if the first term returned by the first example was "eel", ask for the term "eel" and the 24 terms that precede it. From a user perspective, this would be a "page up" from the original list through the ordered list of title terms*
    - http://myserver.com/scan?scanClause=dc.title = eel &responsePosition=25&maximumTerms=25

4 Scan Response

4.1 Summary of Response Elements

The response elements are summarized in the following table.

Table 2. Summary of Response Elements.

Name	Type	Occurrence	Description and/or Reference
<terms>	sequence of <term>	optional	A sequence of <term> elements which match the request. See Term.
<diagnostics>	sequence of <diagnostic>	Optional	A sequence of diagnostics generated during execution. See Diagnostics.
<extraResponseData>	xmlFragment	Optional	Additional information returned by the server. See Extensions.
<echoedScanRequest>	<echoedScanRequest>	Optional	The request parameters echoed back to the client in a simple XML form. See Echoed Request.

4.2 Term

The element <term> has the following subelements.

Table 3. Subelements of element <term>

Name	Type	Occurrence	Description
<value>	xs:string	mandatory	The term, exactly as it appears in the index.
<numberOfRecords>	xs:nonNegativeInteger	optional	The number of records which would be matched if the scanClause were to be searched, with the value of the <value> element for this term substituted for the term in the scanClause.
<displayTerm>	xs:string	optional	A string to display to the end user in place of (or in addition to) the term itself.
<whereInList>	xs:string	optional	See whereInList.
<requestURL>	xs:anyURI	optional	A URL that may be used to send a search for this term to the server.
<extraTermData>	xmlFragment	optional	Additional information concerning the term. See Extensions.

4.3 whereinList

whereInList is provided as a hint about where a term occurs in an ordered index. Often, its presence will explain why fewer terms were returned than requested, but it will also occur when the requested number of terms has been returned.

The sublement <whereInList> (subelement of <term>) is optional; if supplied it MUST have one of the following values:

- 'first' (the first term in the index)
- 'last' (the last term in the index),
- 'only' (the only term in the index)
- 'inner' (an interior term)

4.4 Example Scan Response

```
scanResponse xmlns="http://docs.oasis-open.org/ns/search-ws/sru-2-0-response"
xmlns:xsi=http://www.w3.org/2001/XMLSchema-instance
 xmlns:scan="http://docs.oasis-open.org/ns/search-ws/scan"
xsi:schemaLocation=
"http://docs.oasis-open.org/ns/search-ws/sru-2-0-response
 sru-2-0-response.xsd">
        <scan:terms>
                <scan:term>
                        <scan:value>cartesian</scan:value>
                        <scan:numberOfRecords>35645</scan:numberOfRecords>
                        <scan:displayTerm>Carthesian</scan:displayTerm>
                        <scan:requestURL>
              http://myserver.com/sru/search?query=dc.title%3d%22cartesian%22
          </scan:requestURL>
                </scan:term>
                <scan:term>
                        <scan:value>carthesian</scan:value>
                        <scan:numberOfRecords>2154</scan:numberOfRecords>
                        <scan:displayTerm>CarthÉsian</scan:displayTerm>
                        <scan:requestURL>
            http://myserver.com/sru/search?query=dc.title%3d%22carthesian%22
        </scan:requestURL>
                </scan:term>
                <scan:term>
                        <scan:value>cat</scan:value>
                        <scan:numberOfRecords>8739972</scan:numberOfRecords>
                        <scan:displayTerm>Cat</scan:displayTerm>
                        <scan:requestURL>
              http://myserver.com/sru/search?query=dc.title%3d%22cat%22
        </scan:requestURL>
                </scan:term>
                <scan:term>
                        <scan:value>catholic</scan:value>
                        <scan:numberOfRecords>35</scan:numberOfRecords>
                        <scan:displayTerm>Catholic</scan:displayTerm>
                        <scan:whereInList>last</scan:whereInList>
                        <scan:requestURL>
             http://myserver.com/sru/search?query=dc.title%3d%22catholic%22
        </scan:requestURL>
                </scan:term>
        </scan:terms>
```



```
        <scan:echoedScanRequest>
                <scan:scanClause>dc.title="cat"</scan:scanClause>
                <scan:responsePosition>3</scan:responsePosition>
                <scan:maximumTerms>3</scan:maximumTerms>
                <scan:stylesheet>http://myserver.com/myStyle</scan:stylesheet>
        </scan:echoedScanRequest>
</scanResponse>
```

4.5 Diagnostics

See diagnostic list in the SRU specification.

4.6 Echoed Request

Very thin clients, such as a web browser with a stylesheet, may not have the facility to have recorded the request that generated the response it has just received. The server may thus echo the request back to the client via the response element <**echoedScanRequest**>. There are no request elements associated with this functionality, the server may choose to include it or not within a response.

<echoedSearchRetrieveRequest> includes subelements corresponding to request parameters, using the same name.

Echoed Request Example

```
<echoedScanRequest>
    <scanClause>dc.title = dinosaur</scanClause>
    <startPosition>1</startPosition>
    <maximumTerms>20</maximumTerms>
</echoedSearchRetrieveRequest>
```


5 Extensions

Both in the request and in the response, additional information may be provided - in the request by an extension parameter (whose name is constructed as described next) and in the response by the <**extraResponseData**> and <**extraTermData**> elements.

5.1 Extension Request Parameter

An extension parameter takes on the name of the extension. It must begin with 'x-' : lower case x followed by hyphen. (Scan will never define a parameter with a name beginning with 'x-').

The extension definition MUST supply a namespace. It is recommended that the extension name be 'x-' followed by an identifier for the namespace, again followed by a hyphen, followed by the name of the element within the namespace.

```
 example
http://z3950.loc.gov:7090/voyager?...&x-info4-onScanFail=search
```

Note that this convention does not guarantee uniqueness since the extension name will not include a full URI. The extension owner should try to make the name as unique as possible. If the namespace is identified by an 'info:srw' URI , then the recommended convention is to name the extension "x-infoNNN-XXX" where NNN is the 'info:srw' authority string, and XXX is the name of the extension. Extension names MUST never be assigned with this form except by the proper authority for the given 'info' namespace.

5.2 Extension Response Elements: extraResponseData and extraTermData

An extension definition may (but need not) define a response, to be carried via the extraResponseData and/or extraTermData elements. The extension definition indicates the element names, from the extension's namespace, which will carry the response information.

```
 example:
<sru:extraResponseData>
 <auth:token xmlns:auth="info:srw/extension/2/auth-1.0">
 277c6d19-3e5d-4f2d-9659-86a77fb2b7c8
 </auth:token>
 </sru:extraResponseData>
```

5.3 Behavior

The response may include extraResponseData and/or extraTermData for a given extension only if the request included the extension parameter for that extension, and the extension definition prescribes a response. Thus, a Scan response may never include unsolicited extraResponseData or extraTermData. For example the response may contain cost information regarding the operation on the server or database supplying the results. This data must, however, have been requested.

If the server does not recognize an extension supplied in an extension parameter, it may simply ignore it. (For that matter, even if the server *does* recognize the extension, it may choose to ignore it.) If the particular request requires some confirmation that it has been carried out rather than ignored, then the extension designer should define a response. There may even be an element defined in the response for the server to indicate that it did recognize the request but did not carry it out (and even an indication why). However, the server is never obliged to include a response. Thus though a response may be included in the definition of an extension, it may never be designated as mandatory.

Thus, the semantics of parameters in the request may not be modified by extensions, because the client cannot be assured that the server recognizes the extension. On the other hand, the semantics of parts of the response may be modified by extensions, because the client will be aware that the extension has been invoked, because extensions are always invoked by the client: the response semantics may be changed by an extension only if the client specifically requests the change. Even when a client does request a change in response semantics, it should be prepared to receive regular semantics since servers are at liberty to ignore extensions.

5.4 Echoing the Extension Request

If the server chooses to echo the request to transform the extension parameter into XML, properly namespaced (the extension parameter name will not transform to a valid element in the SRU namespace). If it encounters an unrecognized element and cannot determine the namespace, the server may either make its best guess as to how to transform the element, or simply not return it at all. It should not, however, add an undefined namespace to the element as this would invalidate the response.

6 Conformance

An SRU 2.0 client or server conforms to this standard if it meets the conditions specified in Client Conformance or Server Conformance respectively.

6.1 Client Conformance

6.1.1 Protocol

The client must implement the protocol model. It must support at least one LLP.
The Scan/C must be able to:

1. Accept a request from the CA.
2. Assign values to parameters and form Scan requests according to the procedures described in the standard.
3. Compose an REQ and pass it to HTTP.
4. Accept an RES from HTTP.
5. Decompose the RES and present information from it to the CA.

6.1.2 scanClause

The client must be capable of sending a scanClause.

6.1.3 Response Format

The client must support the 'application/sru+xml' media type for the response.

6.1.4 Diagnostics

The client must support the diagnostic schema and be able to present diagnostics received in an RES to the CA.

6.1.5 Explain

The client must be able to retrieve the Explain record.

6.2 Server Conformance

6.2.1 Protocol

The server must implement the protocol model; it must support at least one LLP.
The Scan/S must be able to:

1. Accept an REQ from HTTP.
2. Decompose the REQ to determine parameter values and interact with the SE as necessary in order to process the request.
3. Assign values to elements and compose an REQ according to the procedures described in the standard.
4. Pass the response to HTTP.

6.2.2 scanClause

The server must support CQL to the extent that it supports a scanClause: it must be able to parse a search clauses consisting of 'index relation searchTerm'.

6.2.3 Response Format

The server must support Application/sru+xml for the response.

6.2.4 Diagnostics

The server must support the diagnostic schema and be able to present diagnostic information received from the SE.

6.2.5 Explain

The Explain record describing the server must be available at the base URL.

Appendix A. Acknowledgements

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. Bindings to Lower Level Protocol (Normative)

B.1 Binding to HTTP GET

This annex describes the construction of a Scan 2.0 http: URL to encode parameter values of the form 'key=value'. Support for Unicode characters is described.

B.1.1 Syntax

The client sends a request via the HTTP GET method. The request is a URI as described in RFC 3986. Specifically it is an HTTP URL of the form:

```
<base URL>?<querystring>
```

using the standard &-separated key=value encoding for parameters in <querystring>.

Example

```
Assume:
 -   The base URL is 'z3950.loc.gov:7090'.
 -   The value of parameter 'scanClause is "title exact dinosaur".

Then the URL would be:
http://z3950.loc.gov:7090/voyager?scanClause=title+exact+dinosaur

And over the wire goes:
GET /voyager?scanClause=title+exact+dinosaur HTTP/1.1
Host: z3950.loc.gov:7090
```

B.1.2 Encoding (Client Procedure)

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 the value to UTF-8.
2. Percent-encode characters as necessary within the value. See RFC 3986 section 2.1.
3. Construct a URI from the parameter names and encoded values.

Note: In step 2, it is recommended to percent-encode every character in a 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.

B.1.3 Decoding (Server Procedure)

1. Parse received request based on '?', '&', and '=' into component parts: the base URL, and parameter names and values.
2. For each parameter:
    a. Decode all %-escapes.
    b. Treat the result as a UTF-8 string.

B.1.4 Example

Consider the following parameter:

scanClause=dc.title = kirkegård

The name of the parameter is "scanClause" and the value is "dc.title = kirkegård"

Note that the first '=' (following "scanClause") must not be percent encoded as it is used as a URI delimiter; it is not part of a parameter name or value. The second '=' must be percent encoded as it is part of a value.

The following characters must be percent encoded:

- the second '=', percent encoded as %3D
- the spaces, 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.

The resulting parameter to be sent to the server would then be:

scanClause=dc.title%20%3D%20kirkeg%C3%A5rd

B.2 Binding to HTTP POST

Rather than construct a URL, the parameters may be sent via POST.

The Content-type header MUST be set to

application/x-www-form-urlencoded'

POST has several benefits over GET. Primarily, the issues with character encoding in URLs are removed, and an explicit character set can be submitted in the Content-type HTTP header. Secondly, very long queries might generate a URL for HTTP GET that is not acceptable by some web servers or client. This length restriction can be avoided by using POST.

The response for SRU via POST is identical to that of SRU via GET.

An example of what might be passed over the wire in the request:

```
POST /voyager HTTP/1.1
Host: z3850.loc.gov:7090
Content-type: application/x-www-form-urlencoded; charset=iso-8859-1
Content-length: 27
scanClause=title%3Ddinosaur
```

B.3 Binding to HTTP SOAP

SRU via SOAP is a binding to the SOAP recommendation of the W3C. The benefits of SOAP are the ease of structured extensions, web service facilities such as proxying and request routing, and the potential for better authentication systems.

In this transport, the request is encoded in XML and wrapped in some additional SOAP specific elements. The response is the same XML as SRU via GET or POST, but wrapped in additional SOAP specific elements.

B.3.1 SOAP Requirements

The specification adheres to the Web Services Interoperability recommendations.

- SOAP version 1.1 is required. V(ersion 1.2 or higher may be supported.
- The service style is 'document/literal'.
- Messages MUST be inline with no multirefs.

- The SOAPAction HTTP header may be present, but should not be required. If present its value MUST be the empty string. It MUST be expressed as:

  SOAPAction: ""

- As specified by SOAP, for version 1.1 the Content-type header MUST be 'text/xml'. For version 1.2 the header value MUST be 'application/soap+xml'. (End points supporting both versions of SOAP as well as SRU via POST thus have three content-type headers to consider.)

B.3.2 Parameter Differences

SRU parameters that cannot be transported via the SOAP binding:

- The 'stylesheet' request parameter MUST NOT be sent. SOAP prevents the use of stylesheets to render the response.

B.3.3 Example SOAP Request

```
<SOAP:Envelope
xmlns:SOAP="http://schemas.xmlsoap.org/soap/envelope/">
  <SOAP:Body>
   <SRW:scanRequest xmlns:SRW="info:srw/xmlns/1/sru ">
    <SRW:scanClause>dinosaur</SRW:query>
    <SRW:startPosition>1</SRW:startRecord>
    <SRW:maximumTerms>20</SRW:maximumRecords>
</SRW:scanRequest>
  </SOAP:Body>
</SOAP:Envelope>
```


Appendix C. Interoperation with Earlier Versions (non-normative)

C.1 Operation and Version

Earlier versions of the protocol (versions 1.1 and 1.2) included request parameters ‘operation’ and ‘version’, and response element <version>. These are removed from version 2.0. This section is included to describe (1) differences imposed by their removal; and (2) how version 2.0 servers may interoperate with clients running earlier versions that include them.

C.1.1 Differences Imposed by their Removal

C.1.1.1 Operation – Request Parameter

Earlier versions as well as this version of Scan express the concept of an operation: a searchRetrieve operation, a scan operation, and an Explain operation are defined. A searchRetrieve or scan request carries a mandatory operation parameter whose value is 'searchRetrieve' or 'scan' respectively, allowing these operations to be distinguished, so that they can both be supported at a single network endpoint.

This specification defines the scan operation and there is also a searchRetrieve operation (a separate specification) however there is no operation parameter for either. As for earlier versions, searchRetrieve and scan may be supported at a single network endpoint because it is heuristically possible to distinguish these operations: a request includes a scanClause parameter if and only if it is a scan request; it contains a query parameter if and only if is a searchRetrieve request. However, if a new operation were to be defined, or a new version of searchRetrieve or scan, then it may no longer be possible to heuristically distinguish these operations and then it may be necessary to define the operation parameter for one or more operations.

C.1.1.2 Version – Request Parameter and Response Element

In earlier versions a version request parameter and response element were defined because it was assumed that multiple versions might be supported at a single endpoint. With version 2.0, the version request parameter and response element are removed and it is EXPLICITY ASSUMED that there will be different endpoints for different versions.

C.1.2 Interoperation

Following are guidelines for interoperation with implementations of earlier versions where the operation or version parameter or element is used.

- A client operating under version 2.0 of Scan SHOULD include NEITHER of the request parameters ‘operation’ or ‘version’.
- **If a server is operating under version 2.0 and the version parameter is included in a received request, and the server supports the requested version**: It may, if it chooses, process the request under that version. However, details of how these parameters are treated are beyond the scope of this standard.
- **If the version parameter is included in a received request, and the server does not supports the requested version**: the request should be rejected and a fatal diagnostic included. (The server may be able to supply the response according to the requested version – even though it does not in general support that version. If the server is willing and able to issue the failure response according to the requested version, then it should do so. However it is not required to do so, and, unfortunately, failure to do will probably mean that the response cannot be interpreted by the client.)
- **If the operation parameter is included in a received request:**

- If the value of the operation parameter is searchRetrieve, the server may ignore it.
- If the value of the operation parameter is other than searchRetrieve, the server may reject the request (with a fatal diagnostic) or may, if it chooses, process the request; however details of how the request should be processed are out of scope.

