



searchRetrieve: Part 6. SRU Scan Operation Version 1.0

Candidate OASIS Standard 01

25 October 2012

Specification URIs

This version:

<http://docs.oasis-open.org/search-ws/searchRetrieve/v1.0/cos01/part6-scan/searchRetrieve-v1.0-cos01-part6-scan.doc> (Authoritative)
<http://docs.oasis-open.org/search-ws/searchRetrieve/v1.0/cos01/part6-scan/searchRetrieve-v1.0-cos01-part6-scan.html>
<http://docs.oasis-open.org/search-ws/searchRetrieve/v1.0/cos01/part6-scan/searchRetrieve-v1.0-cos01-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 Dixon (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/cos01/schemas/>
- *searchRetrieve: Part 0. Overview Version 1.0.*
<http://docs.oasis-open.org/search-ws/searchRetrieve/v1.0/cos01/part0-overview/searchRetrieve-v1.0-cos01-part0-overview.html>
- *searchRetrieve: Part 1. Abstract Protocol Definition Version 1.0.*
<http://docs.oasis-open.org/search-ws/searchRetrieve/v1.0/cos01/part1-apd/searchRetrieve-v1.0-cos01-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/cos01/part2-sru1.2/searchRetrieve-v1.0-cos01-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/cos01/part3-sru2.0/searchRetrieve-v1.0-cos01-part3-sru2.0.html>
- *searchRetrieve: Part 4. APD Binding for OpenSearch Version 1.0.*
<http://docs.oasis-open.org/search-ws/searchRetrieve/v1.0/cos01/part4-opensearch/searchRetrieve-v1.0-cos01-part4-opensearch.html>
- *searchRetrieve: Part 5. CQL: The Contextual Query Language Version 1.0.*
<http://docs.oasis-open.org/search-ws/searchRetrieve/v1.0/cos01/part5-cql/searchRetrieve-v1.0-cos01-part5-cql.html>
- *searchRetrieve: Part 6. SRU Scan Operation Version 1.0.* (this document)
<http://docs.oasis-open.org/search-ws/searchRetrieve/v1.0/cos01/part6-scan/searchRetrieve-v1.0-cos01-part6-scan.html>
- *searchRetrieve: Part 7. SRU Explain Operation Version 1.0.*
<http://docs.oasis-open.org/search-ws/searchRetrieve/v1.0/cos01/part7-explain/searchRetrieve-v1.0-cos01-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. 25 October 2012. Candidate OASIS Standard 01. <http://docs.oasis-open.org/search-ws/searchRetrieve/v1.0/cos01/part6-scan/searchRetrieve-v1.0-cos01-part6-scan.html>.

Notices

Copyright © OASIS Open 2012. All Rights Reserved.

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/policies-guidelines/trademark> for above guidance.

Table of Contents

1	Introduction.....	5
1.1	Terminology.....	5
1.2	References.....	5
1.3	Namespace.....	5
2	Overview and Model.....	6
2.1	Operation Model.....	6
2.2	Data model.....	6
2.3	Protocol Model.....	6
2.4	Processing Model.....	7
2.5	Query model.....	7
2.6	Diagnostic Model.....	8
2.7	Explain Model.....	8
2.8	Serialization Model.....	8
3	Scan Request.....	9
3.1	Summary of Request Parameters.....	9
3.2	Request Parameter Descriptions.....	9
3.3	Serialization of Request Parameters.....	11
4	Scan Response.....	12
4.1	Summary of Response Elements.....	12
4.2	Term.....	12
4.3	whereInList.....	12
4.4	Example Scan Response.....	13
4.5	Diagnostics.....	14
4.6	Echoed Request.....	14
5	Extensions.....	15
5.1	Extension Request Parameter.....	15
5.2	Extension Response Elements: extraResponseData and extraTermData.....	15
5.3	Behavior.....	15
5.4	Echoing the Extension Request.....	16
6	Conformance.....	17
6.1	Client Conformance.....	17
6.2	Server Conformance.....	17
Appendix A.	Acknowledgements.....	19
Appendix B.	Bindings to Lower Level Protocol (Normative).....	20
B.1	Binding to HTTP GET.....	20
B.2	Binding to HTTP POST.....	21
B.3	Binding to HTTP SOAP.....	21
Appendix C.	Interoperation with Earlier Versions (non-normative).....	23
C.1	Operation and Version.....	23

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>

29 2 Overview and Model

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

35 2.1 Operation Model

36 The SWS initiative defines three operations:

- 37 1. **SearchRetrieve Operation.** The main operation. The SRU protocol defines a request message
38 (sent from an SRU client to an SRU server) and a response message (sent from the server to the
39 client). This transmission of an SRU request followed by an SRU response constitutes a
40 SearchRetrieve operation.
- 41 2. **Scan Operation.** The Scan operation is defined by the Scan protocol, which is this specification.
42 Similar to SRU, it defines a request message and a response message. The transmission of a
43 Scan request followed by a Scan response constitutes a Scan operation.
- 44 3. **Explain Operation.** See [Explain Model](#). When a client retrieves an Explain record, this
45 constitutes an Explain operation.

46 *Note: In earlier versions a searchRetrieve or scan request carried a mandatory operation parameter. In*
47 *version 2.0, there is no operation parameter for either. See [Interoperation with Earlier Versions](#).*

48 2.2 Data model

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

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

56 2.3 Protocol Model

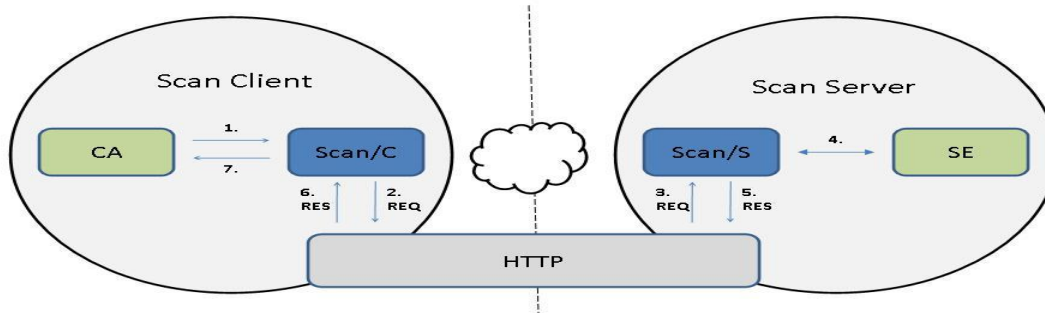
57 The protocol model assumes these conceptual components:

- 58 - The client application (**CA**),
- 59 - the Scan protocol module at the client (**Scan/C**),
- 60 - the lower level protocol (**HTTP**),
- 61 - the Scan protocol module at the server (**Scan/S**),
- 62 - the search engine at the server (**SE**).

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

- 67 • At the client system the Scan/C accepts a request from the CA, formulates a searchRetrieve
68 protocol request (**REQ**) and passes it to HTTP.
- 69 • Subsequently at the server system HTTP passes the request to the Scan/S which interacts with
70 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:



- CA passes a request to Scan/C.
- Scan/C formulates a REQ and passes it to HTTP.
- HTTP passes the REQ to Scan/S.
- Scan/S interacts with SE to form a RES.
- The RES is passed to HTTP.
- HTTP passes the RES to Scan/C.
- 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.

100 **2.6 Diagnostic Model**

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

103 **2.7 Explain Model**

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

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

113 **2.8 Serialization Model**

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

118 3 Scan Request

119 3.1 Summary of Request Parameters

120 The request parameters are summarized in the following table.

121 *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

122 3.2 Request Parameter Descriptions

123 3.2.1 scanClause

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

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

128 For example, the scanClause "title==cat" indicates the index 'title' and start term 'cat'.

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

132 3.2.2 responsePosition and maximumTerms

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

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

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

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

143 The first term in the list is determined as follows.

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

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

152 The actual number of terms supplied in the list SHOULD NOT exceed M, but may be fewer, for example if
153 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

154

155 3.2.3 httpAccept

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

159 The default value for the response type is '[application/sru+xml](#)'.

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

162 The intent of the httpAccept parameter can be accomplished with an HTTP Accept header. Servers
163 SHOULD support either mechanism. In either case (via the httpAccept parameter or HTTP Accept
164 header), if the server does not support the requested media type then the server MUST respond with a
165 406 status code and SHOULD return an HTML message with pointers to that resource in supported
166 media types.

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

170 For instance, if the client had sent the URL

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

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

173 then the server SHOULD return a Content-Location value of

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

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

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

182 <http://example.org/scan?query=dog>

183 a Content-Location header of

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

185 should be returned.

186 3.2.4 Stylesheet

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

191 The purpose is to allow a thin client to turn the response XML into a natively renderable format, often
192 HTML or XHTML. This allows a web browser or other application capable of rendering stylesheets, to act
193 as a dedicated client without requiring any further application logic.

194 **Example**

195 <http://z3950.loc.gov:7090/voyager?stylesheet=/master.xsl&query=dinosaur>

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

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

200 3.3 Serialization of Request Parameters

- 201 • **Example:** *Get 25 title terms centered on the word "frog"*

- 202 ○ <http://myserver.com/scan?scanClause=dc.title = frog>
203 &responsePosition=13&maximumTerms=25

- 204 • **Example:** *If the last term returned by the first example was "goat", ask for the term "goat" and the*
205 *24 terms that follow it. From a user perspective, this would be a "page down" through the ordered*
206 *list of title terms*

- 207 ○ <http://myserver.com/scan?scanClause=dc.title = goat>
208 &responsePosition=1&maximumTerms=25

- 209 • **Example:** *if the first term returned by the first example was "eel", ask for the term "eel" and the 24*
210 *terms that precede it. From a user perspective, this would be a "page up" from the original list*
211 *through the ordered list of title terms*

- 212 ○ <http://myserver.com/scan?scanClause=dc.title = eel>
213 &responsePosition=25&maximumTerms=25

214 4 Scan Response

215 4.1 Summary of Response Elements

216 The response elements are summarized in the following table.

217 *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 .

218 4.2 Term

219 The element <term> has the following subelements.

220 *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 .

221 4.3 whereinList

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

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

- 227 • 'first' (the first term in the index)
- 228 • 'last' (the last term in the index),
- 229 • 'only' (the only term in the index)
- 230 • 'inner' (an interior term)

231 4.4 Example Scan Response

```

232 scanResponse xmlns="http://docs.oasis-open.org/ns/search-ws/sru-2-0-response"
233 xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance
234 xmlns:scan="http://docs.oasis-open.org/ns/search-ws/scan"
235 xsi:schemaLocation=
236 "http://docs.oasis-open.org/ns/search-ws/sru-2-0-response
237 sru-2-0-response.xsd">
238   <scan:terms>
239     <scan:term>
240       <scan:value>cartesian</scan:value>
241       <scan:numberOfRecords>35645</scan:numberOfRecords>
242       <scan:displayTerm>Carthesian</scan:displayTerm>
243       <scan:requestURL>
244         http://myserver.com/sru/search?query=dc.title%3d%22cartesian%22
245       </scan:requestURL>
246     </scan:term>
247     <scan:term>
248       <scan:value>carthesian</scan:value>
249       <scan:numberOfRecords>2154</scan:numberOfRecords>
250       <scan:displayTerm>CarthÉsian</scan:displayTerm>
251       <scan:requestURL>
252         http://myserver.com/sru/search?query=dc.title%3d%22carthesian%22
253       </scan:requestURL>
254     </scan:term>
255     <scan:term>
256       <scan:value>cat</scan:value>
257       <scan:numberOfRecords>8739972</scan:numberOfRecords>
258       <scan:displayTerm>Cat</scan:displayTerm>
259       <scan:requestURL>
260         http://myserver.com/sru/search?query=dc.title%3d%22cat%22
261       </scan:requestURL>
262     </scan:term>
263     <scan:term>
264       <scan:value>catholic</scan:value>
265       <scan:numberOfRecords>35</scan:numberOfRecords>
266       <scan:displayTerm>Catholic</scan:displayTerm>
267       <scan:whereInList>last</scan:whereInList>
268       <scan:requestURL>
269         http://myserver.com/sru/search?query=dc.title%3d%22catholic%22
270       </scan:requestURL>
271     </scan:term>
272   </scan:terms>

```

```
273     <scan:echoedScanRequest>
274         <scan:scanClause>dc.title="cat"</scan:scanClause>
275         <scan:responsePosition>3</scan:responsePosition>
276         <scan:maximumTerms>3</scan:maximumTerms>
277         <scan:stylesheet>http://myserver.com/myStyle</scan:stylesheet>
278     </scan:echoedScanRequest>
279 </scanResponse>
```

280 4.5 Diagnostics

281 See diagnostic list in the SRU specification.

282 4.6 Echoed Request

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

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

289 Echoed Request Example

```
290
291 <echoedScanRequest>
292     <scanClause>dc.title = dinosaur</scanClause>
293     <startPosition>1</startPosition>
294     <maximumTerms>20</maximumTerms>
295 </echoedSearchRetrieveRequest>
```

296
297

298 5 Extensions

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

302 5.1 Extension Request Parameter

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

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

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

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

316 5.2 Extension Response Elements: extraResponseData and 317 extraTermData

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

321 **example:**
322 `<sru:extraResponseData>`
323 `<auth:token xmlns:auth="info:srw/extension/2/auth-1.0">`
324 `277c6d19-3e5d-4f2d-9659-86a77fb2b7c8`
325 `</auth:token>`
326 `</sru:extraResponseData>`

327 5.3 Behavior

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

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

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

347 **5.4 Echoing the Extension Request**

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

353

354 6 Conformance

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

357 6.1 Client Conformance

358 6.1.1 Protocol

359 The client must implement the [protocol model](#). It must support at least one LLP.
360 The Scan/C must be able to:

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

367 6.1.2 scanClause

368 The client must be capable of sending a scanClause.

369 6.1.3 Response Format

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

371 6.1.4 Diagnostics

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

374 6.1.5 Explain

375 The client must be able to retrieve the Explain record.

376 6.2 Server Conformance

377 6.2.1 Protocol

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

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

386 **6.2.2 scanClause**

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

389 **6.2.3 Response Format**

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

391 **6.2.4 Diagnostics**

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

394 **6.2.5 Explain**

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

396 **Appendix A. Acknowledgements**

397 Acknowledgements are supplied in the Overview document:

398 *searchRetrieve: Part 0. Overview Version 1.0*

399 <http://docs.oasis-open.org/search-ws/searchRetrieve/v1.0/csd01/part0-overview/searchRetrieve->
400 [v1.0-csd01-part0-overview.doc](http://docs.oasis-open.org/search-ws/searchRetrieve/v1.0/csd01/part0-overview/searchRetrieve-v1.0-csd01-part0-overview.doc)

401 Appendix B. Bindings to Lower Level Protocol 402 (Normative)

403 B.1 Binding to HTTP GET

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

406 B.1.1 Syntax

407 The client sends a request via the HTTP GET method. The request is a URI as described in [RFC 3986](#).
408 Specifically it is an HTTP URL of the form:

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

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

411 Example

412 Assume:

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

415

416 Then the URL would be:

```
417 http://z3950.loc.gov:7090/voyager?scanClause=title+exact+dinosaur
```

418

419 And over the wire goes:

```
420 GET /voyager?scanClause=title+exact+dinosaur HTTP/1.1
```

```
421 Host: z3950.loc.gov:7090
```

422 B.1.2 Encoding (Client Procedure)

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

- 425 1. Convert the value to UTF-8.
- 426 2. Percent-encode characters as necessary within the value. See [RFC 3986](#) section 2.1.
- 427 3. Construct a URI from the parameter names and encoded values.

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

433 B.1.3 Decoding (Server Procedure)

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

439 B.1.4 Example

440 Consider the following parameter:

441 scanClause=dc.title = kirkegård
442 The name of the parameter is "scanClause" and the value is "dc.title = kirkegård"
443 Note that the first '=' (following "scanClause") must not be percent encoded as it is used as a URI
444 delimiter; it is not part of a parameter name or value. The second '=' must be percent encoded as it is part
445 of a value.

446 The following characters must be percent encoded:

- 447 • the second '=', percent encoded as %3D
- 448 • the spaces, percent encoded as %20
- 449 • the 'å'. Its UTF-8 representation is C3A5, two octets, and correspondingly it is represented in a
450 URI as two characters percent encoded as %C3%A5.

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

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

453 **B.2 Binding to HTTP POST**

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

455 The Content-type header MUST be set to

456 **application/x-www-form-urlencoded'**

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

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

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

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

468 **B.3 Binding to HTTP SOAP**

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

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

475 **B.3.1 SOAP Requirements**

476 The specification adheres to the [Web Services Interoperability](#) recommendations.

- 477 • SOAP version 1.1 is required. Version 1.2 or higher may be supported.
- 478 • The service style is 'document/literal'.
- 479 • Messages MUST be inline with no multirefs.

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

482 **SOAPAction:** ""

- 483 • As specified by SOAP, for version 1.1 the Content-type header MUST be 'text/xml'. For version
484 1.2 the header value MUST be 'application/soap+xml'. (End points supporting both versions of
485 SOAP as well as SRU via POST thus have three content-type headers to consider.)

486 **B.3.2 Parameter Differences**

487 SRU parameters that cannot be transported via the SOAP binding:

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

490 **B.3.3 Example SOAP Request**

```
491           <SOAP:Envelope  
492           xmlns:SOAP="http://schemas.xmlsoap.org/soap/envelope/">  
493           <SOAP:Body>  
494           <SRW:scanRequest xmlns:SRW="info:srw/xmlns/1/sru ">  
495           <SRW:scanClause>dinosaur</SRW:query>  
496           <SRW:startPosition>1</SRW:startRecord>  
497           <SRW:maximumTerms>20</SRW:maximumRecords>  
  
498           </SRW:scanRequest>  
499           </SOAP:Body>  
500           </SOAP:Envelope>
```

501

502 Appendix C. Interoperation with Earlier Versions (non- 503 normative)

504 C.1 Operation and Version

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

509 C.1.1 Differences Imposed by their Removal

510 C.1.1.1 Operation – Request Parameter

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

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

523 C.1.1.2 Version – Request Parameter and Response Element

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

528 C.1.2 Interoperation

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

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

- 545
- 546
- 547
- 548
- 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.