



searchRetrieve: Part 2. searchRetrieve Operation: APD Binding for SRU 1.2 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/part2-sru1.2/searchRetrieve-v1.0-cos01-part2-sru1.2.doc> (Authoritative)
<http://docs.oasis-open.org/search-ws/searchRetrieve/v1.0/cos01/part2-sru1.2/searchRetrieve-v1.0-cos01-part2-sru1.2.html>
<http://docs.oasis-open.org/search-ws/searchRetrieve/v1.0/cos01/part2-sru1.2/searchRetrieve-v1.0-cos01-part2-sru1.2.pdf>

Previous version:

N/A

Latest version:

<http://docs.oasis-open.org/search-ws/searchRetrieve/v1.0/searchRetrieve-v1.0-part2-sru1.2.doc>
(Authoritative)
<http://docs.oasis-open.org/search-ws/searchRetrieve/v1.0/searchRetrieve-v1.0-part2-sru1.2.html>
<http://docs.oasis-open.org/search-ws/searchRetrieve/v1.0/searchRetrieve-v1.0-part2-sru1.2.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.* (this document)
<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.*
<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:

This specification is related to:

- Search/Retrieval via URL. The Library of Congress. <http://www.loc.gov/standards/sru/>

Abstract:

This document specifies a binding of the OASIS SWS Abstract Protocol Definition to the specification of version 1.2 of the protocol SRU: Search/Retrieve via URL. This is one of a set of documents for the OASIS Search Web Services (SWS) initiative.

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:

[SearchRetrievePt2]

searchRetrieve: Part 2. searchRetrieve Operation: APD Binding for SRU 1.2 Version 1.0. 25 October 2012. Candidate OASIS Standard 01. <http://docs.oasis-open.org/search-ws/searchRetrieve/v1.0/cos01/part2-sru1.2/searchRetrieve-v1.0-cos01-part2-sru1.2.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.....	6
1.1	Terminology.....	6
1.2	References.....	6
1.3	Namespace.....	6
2	Overview and Model.....	7
2.1	Relationship to Abstract Protocol Definition.....	7
2.2	Operation Model.....	7
2.3	Data model.....	7
2.4	Protocol Model.....	8
2.5	Query model.....	9
2.6	Result Set Model.....	9
2.7	Diagnostic Model.....	10
2.8	Explain Model.....	10
3	Request Parameters (Summary).....	11
3.1	Actual Request Parameters for this Binding.....	11
3.2	Relationship of Actual Parameters to Abstract Parameters.....	11
4	Response Elements (Summary).....	13
4.1	Actual Response Elements for this Binding.....	13
4.2	Relationship of Actual Elements to Abstract Elements.....	13
5	Parameter and Element Descriptions.....	15
5.1	startRecord and maximumRecords.....	15
5.2	resultSetTTL, and resultSetIdleTime.....	15
5.3	numberOfRecords.....	15
5.4	nextRecordPosition.....	15
5.5	resultSetId.....	16
5.6	Echoed Request.....	16
6	Record Serialization and formatting Parameters.....	17
6.1	recordPacking.....	17
6.2	recordSchema.....	17
6.3	Stylesheet.....	17
7	Response Records.....	18
	Example.....	18
8	Diagnostics.....	20
8.1	Diagnostic List.....	20
8.2	Diagnostic Data Elements.....	20
8.3	Diagnostic Examples.....	20
	8.3.1 Non-Surrogate Example.....	20
	8.3.2 Surrogate Example.....	21
9	Extensions.....	22
9.1	Extension Request Parameter.....	22
9.2	Extension Response Element: extraResponseData.....	22
9.3	Behavior.....	22
9.4	Echoing the Extension Request.....	23

10	Conformance	24
10.1	Client Conformance	24
10.1.1	Protocol	24
10.1.2	Query	24
10.1.3	Response Format.....	24
10.1.4	Diagnostics.....	24
10.1.5	Explain	24
10.2	Server Conformance.....	24
10.2.1	Protocol	24
10.2.2	Query	24
10.2.3	Response Format.....	25
10.2.4	Diagnostics.....	25
10.2.5	Explain	25
Appendix A.	Acknowledgements	26
Appendix B.	SRU 1.2 Bindings to Lower Level Protocol	27
B.1	Binding to HTTP GET.....	27
B.1.1	Syntax	27
B.1.2	Encoding (Client Procedure).....	27
B.1.3	Decoding (Server Procedure)	27
B.1.4	Example	28
B.2	Binding to HTTP POST	28
B.3	Binding to HTTP SOAP	28
B.3.1	SOAP Requirements.....	28
B.3.2	Parameter Differences	29
B.3.3	Example SOAP Request.....	29
B.3.4	WSDL.....	29
Appendix C.	Diagnostics for use with SRU 1.2	30
C.1	Notes	32

1 Introduction

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

This document, “searchRetrieve: Part 1. SearchRetrieve Operation: Binding for SRU 1.2” is the specification of version 1.2 of the protocol SRU: Search/Retrieve via URL.

This specification is intended to be compatible with the specification at <http://www.loc.gov/standards/sru/specs/>

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

The collection of documents also includes three bindings. This document is one of the three.

Scan, a companion protocol to SRU, supports index browsing, to help a user formulate a query. The Scan specification is also one of the documents in this collection.

Finally, the Explain specification, also in this collection, describes a server’s Explain file, which provides information for a client to access, query and process results from that server.

The set of documents in this collection of specifications are:

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

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>

40 2 Overview and Model

41 2.1 Relationship to Abstract Protocol Definition

42 The APD defines abstract request parameters and abstract response elements. A binding lists those
43 abstract parameters and elements applicable to that binding and indicates the corresponding actual name
44 of the parameter or element to be transmitted in a request or response.

45 **Example.**

46 The APD defines the abstract parameter: `startPosition` as "The position within
47 the result set of the first item to be returned. "
48 And this specification refers to that abstract parameter and notes that its
49 name, as used in this specification is 'startRecord'. Thus the request
50 parameter 'startRecord' in this specification represents the abstract
51 parameter `startPosition` in the APD.

52 Different bindings may use different names to represent this same abstract parameter, and its semantics
53 may differ across those bindings as the binding models differ. It is the responsibility of the binding to
54 explain these differences in terms of their respective models.

55 2.2 Operation Model

56 This specification defines the protocol **SRU: Search/Retrieve via URL**. Different bindings may define
57 different protocols for search/retrieve. The SRU protocol defines a request message (sent from an SRU
58 client to an SRU server) and a response message (sent from the server to the client). This transmission of
59 an SRU request followed by an SRU response is called a *SearchResponse operation*.

60 For the SRU protocol, three operations are defined:

- 61 1. **SearchResponse Operation.** The SearchResponse operation is defined by the SRU protocol,
62 which is this specification.
- 63 2. **Scan Operation.**
- 64 3. **Explain Operation.** See [Explain Model](#) below.

65 2.3 Data model

66 A server exposes a database for access by a remote client for purposes of search and retrieval. The
67 database is a collection of units of data, each referred to as an *abstract record*. In this model there is a
68 single database at any given server.

69 Associated with a database are one or more formats that the server may apply to an abstract record,
70 resulting in an exportable structure referred to as a *response record*.

71 Note:

72 The term *record* is often used in place of "abstract record" or "response record" when the meaning
73 is clear from the context or when the distinction is not important.

74 Such a format is referred to as a *record schema*. It represents a common understanding shared by the
75 client and server of the information contained in the records of the database, to allow the transfer of that
76 information. It does not represent nor does it constrain the internal representation or storage of that
77 information at the server.

78

Relationship of Data Model to Abstract Model

The data model in the APD says that a "datastore is a collection of units of data. Such a unit is referred to as an abstract item...".

In this binding:

- A datastore is referred to as a database.
- An item is referred to as a record.

The APD further notes that “Associated with a datastore are one or more formats that the server may apply to an abstract item, resulting in an exportable structure referred to as a response Item. Such a format is referred to as a response item type or item type.”

In this Binding:

- An item type is referred to as a record schema.

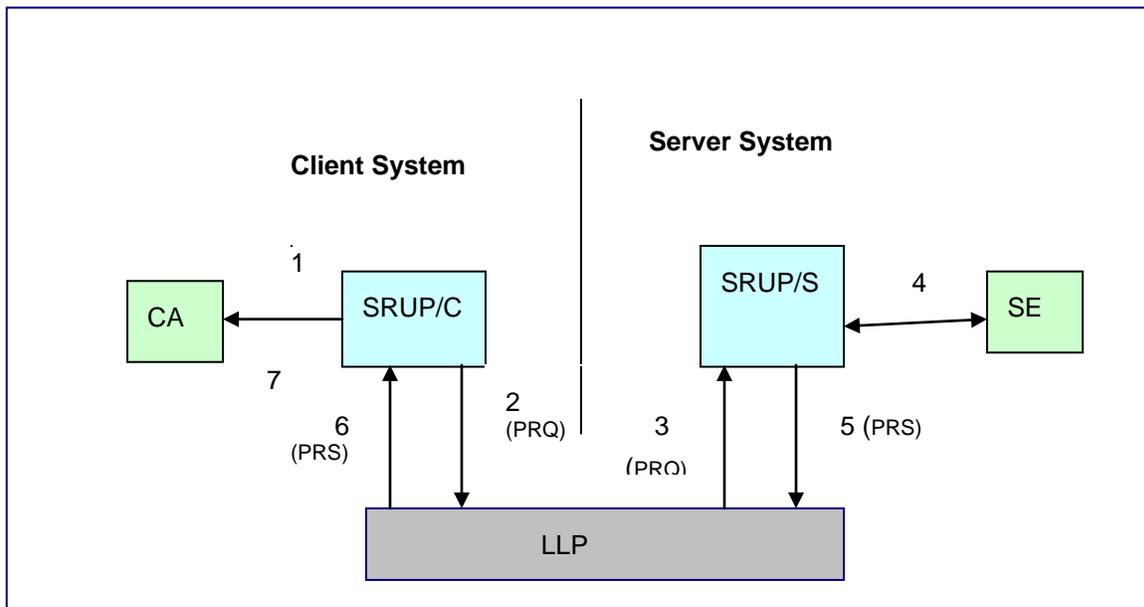
79 2.4 Protocol Model

80 The protocol model assumes these conceptual components:

- 81 - The client application (CA),
- 82 - the SRU protocol module at the client (SRUP/C),
- 83 - the Lower Level Protocol (LLP)
- 84 - the SRU protocol module at the server (SRUP/S),
- 85 - the search engine at the server (SE).

86 For modeling purposes this standard assumes but does not prescribe bindings between the CA and
87 SRUP/C and between SRUP/S and SE, as well as between each of SRUP/C and SRUP/S and LLP; for
88 examples of the latter two see [Bindings to Lower Level Protocols](#). The conceptual model of protocol
89 interactions is as follows:

- 90 • At the client system the SRUP/C accepts a request from the CA, formulates a searchRetrieve
91 protocol request (PRQ) and passes it to the LLP.
- 92 • Subsequently at the server system the LLP passes the request to the SRUP/S which interacts
93 with the SE, forms a searchRetrieve protocol response (PRS), and passes it to the LLP.
- 94 • At the client system, the LLP passes the response to the SRUP/C which presents results to the
95 CA.



96 The protocol model is described diagrammatically in this picture.

97

- 98 1. CA passes a request to SRUP/C.
- 99 2. SRUP/C formulates a PRQ and passes it to LLP.

- 100 3. LLP passes the PRQ to SRUP/S.
- 101 4. SRUP/S interacts with SE to form a PRS.
- 102 5. The PRS is passed to the LLP.
- 103 6. LLP passes the PRS to SRUP/C.
- 104 7. SRUP/C presents results to CA
- 105 Processing Model

106 A client sends a searchRetrieve request to a server. The request includes request parameters including
107 a query to be matched against the database at the server. The server processes the query, creating a
108 result set of records that match the query.

109 The request also indicates the desired number of records to be included in the response and includes the
110 identifier of a record schema for transfer of the records in the response, as well as the identifier of a
111 response schema for transfer of the entire response (including all of the response records).

112 The response includes records from the result set, diagnostic information, and a result set identifier that
113 the client may use in a subsequent request to retrieve additional records.

114 2.5 Query model

115 The query language to be used for SRU version 1.2 is the Contextual Query Language, CQL. The
116 following is intended as only a very cursory overview of CQL's capabilities; for details, consult the CQL
117 specification.

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

126 2.6 Result Set Model

127 This is a logical model; support of result sets is neither assumed nor required by this standard. There are
128 applications where result sets are critical and applications where result sets are not viable.

129 When a query is processed, a set of matching records is selected and that set is represented by a result
130 set maintained at the server. The result set, logically, is an ordered list of references to the records. Once
131 created, a result set cannot be modified; any process that would somehow change a result set is viewed
132 logically to instead create a new result set. (For example, an existing result set may be sorted. In that
133 case, the existing result set is logically viewed to be deleted, and a new result set – the sorted set -
134 created.) Each result set is referenced via a unique identifying string, generated by the server when the
135 result set is created.

136 From the client point of view, the result set is a set of abstract records each referenced by an ordinal
137 number, beginning with 1. The client may request a given record from a result set according to a specific
138 format. For example the client may request record 1 in the Dublin Core format, and subsequently request
139 record 1 in the MODS [7] format. The format in which records are supplied is not a property of the result
140 set, nor is it a property of the abstract records as a member of the result set; the result set is simply the
141 ordered list of abstract records. How the client references a record in the result set is unrelated to how the
142 server may reference it.

143 The records in a result set are not necessarily ordered according to any specific or predictable scheme.
144 The server determines the order of the result set, unless it has been created with a request that includes
145 a sort specification. (In that case, only the final sorted result set is considered to exist, even if the server
146 internally creates a temporary result set and then sorts it. The unsorted, temporary result set is not
147 considered to have ever existed, for purposes of this model.) In any case, the order must not change. (As
148 noted above, if a result set is created and subsequently sorted, a new result set must be created.)

149 Thus, suppose an abstract record is deleted or otherwise becomes unavailable while a result set which
150 references that record still exists. This MUST not cause re-ordering. For example, if a client retrieves
151 records 1 through 3, and subsequently record 2 becomes unavailable, if the server again requests record
152 3, it must be the same record 3 that was returned as record 3 in the earlier operation. (“Same record”
153 does not necessarily mean the same content; the record’s content may have changed.) If the server
154 requests record 2 (no longer available) the server should supply a surrogate diagnostic (see [Diagnostic
155 Model](#)) in place of the response record for record 2.

Relationship of Result Set Model to Abstract Model

The result set model for SRU 1.2 is as described in the Abstract Protocol Definition, with the following exceptions:

- Addition of the preceding paragraph (beginning with “when a result set record becomes unavailable...”).
- The APD says “A server might support requests by record ... or it may instead support requests by group. It may support one form only or both.” That sentence has been deleted. In SRU requests are by record; groups are not supported.

156

157 2.7 Diagnostic Model

158 A server supplies diagnostics in the response as appropriate. A diagnostics is *fatal* or *non-fatal*. A fatal
159 diagnostic is generated when the execution of the request cannot proceed and no results are available.
160 For example, if the client supplied an invalid query there might be nothing that the server can do. A non-
161 fatal diagnostic is one where processing may be affected but the server can continue. For example if a
162 particular record is not available in the requested schema but others are, the server may return the ones
163 that are available rather than failing the entire request.

164 Non-fatal diagnostics are further divided into two categories: 'surrogate' and 'non-surrogate'. Surrogate
165 diagnostics take the place of a record (as described in the Result Set Model). Non-surrogate, non-fatal
166 diagnostics are diagnostics saying that while some or all the entries are available, something may have
167 gone wrong; for example the requested sorting algorithm might not be available. Or, it may be just a
168 warning. See [Diagnostics](#).

169 2.8 Explain Model

170 Every SRU server provides an associated Explain record. The standard requires that this record be
171 retrievable as the response of an HTTP GET at the base URL for SRU server. The Explain record for a
172 serve may be obtained from other sources as well. An SRU client may retrieve this record which provides
173 information about the server’s capabilities. The client may use the information in the Explain record to
174 self-configure and provide an appropriate interface to the user.

175 The Explain record provides such details as query types supported, CQL context sets (and for each
176 context set indexes supported), diagnostic sets, record schemas, sorting capabilities, specification of
177 defaults, and other details. It also includes sample queries, and conditions of use (for example mandatory
178 display of copyright and syndication rights).

179

180 **3 Request Parameters (Summary)**

181 As noted at the beginning of this document, the APD defines abstract request parameters. A binding,
 182 such as this specification, lists those abstract parameters and indicates the corresponding actual names
 183 of the parameter to be transmitted in a request. Below, the actual parameters for this binding are listed,
 184 and following that, the binding of each parameter to its corresponding abstract parameter in the APD.

185 **3.1 Actual Request Parameters for this Binding**

186 The following table provides a summary of the actual request parameters defined in this binding,.

187 Table 1: *Summary of Request Parameters*

Actual Parameter Name	Occurrence	Reference/Description
operation	Mandatory, non repeatable	The string: 'searchRetrieve'.
version	mandatory, non repeatable	The string '1.2'.
query	Mandatory, non repeatable	A query expressed in CQL .
startRecord	optional, non repeatable	See startRecord and maximumRecords .
maximumRecords	optional, non repeatable	
recordPacking	optional, non repeatable	See recordPacking .
recordSchema	optional, non repeatable	See recordSchema .
resultSetTTL	optional, non repeatable	See resultSetTTL and resultSetIdleTime .
stylesheet	optional, non repeatable	See stylesheet .
Extension Parameters	optional	See Extensions

188 **3.2 Relationship of Actual Parameters to Abstract Parameters**

189 The following table summarizes the relationship of actual parameters to abstract parameters defined in
 190 the APD. In the first two columns are shown abstract parameters and their corresponding actual
 191 parameters for those abstract parameters that have corresponding actual parameters in this binding. The
 192 third column shows abstract parameters for which no corresponding actual parameters are defined for
 193 this binding. The fourth column lists new parameters defined for this binding, that is, for which there are
 194 no corresponding abstract parameters.

195 Table 2: *Relationship of Actual Parameters to Abstract Parameters*

Abstract Parameter	Corresponding Actual Parameter	Excluded Abstract Parameters	Additional Actual Parameters
Query	Query		
startPosition	startRecord		
MaximumItems	maximumRecords		
responseItemTypes	recordSchema		

		SortOrder	
		responseFormat	
		Group	
			operation
			version
			recordPacking
			resultSetTTL
			stylesheet
			<i>Extension parameters</i>

196
197
198
199
200

4 Response Elements (Summary)

The APD defines abstract response elements. Binding list those abstract elements and indicate the corresponding actual names of the parameter to be transmitted in a response. Below, the actual elements for this binding are listed, and following that, the binding of each elements to its corresponding abstract element in the APD.

201
202
203

4.1 Actual Response Elements for this Binding

The following table describes the top-level XML elements in the response.

Table 3: Summary of Actual Response Elements

Actual Element Name	Type	Occurrence	Restrictions/Values/Description
<version>	<i>xs:string</i>	Mandatory, non-repeatable	The string '1.2'.
<numberOfRecords>	<i>xs:integer</i>	Mandatory, non-repeatable	See numberOfRecords
<resultSetId>	<i>xs:string</i>	Optional, non-repeatable	See resultSetId
<resultSetIdleTime>	<i>xs:integer</i>	Optional, non-repeatable	see resultSetTTL and resultSetIdleTime .
<records>	<i>structured</i>	Optional, repeatable	see records
<nextRecordPosition>	<i>xs:integer</i>	Optional, non-repeatable	see nextRecordPosition
<diagnostics>	<i>structured</i>	Optional, non-repeatable	see diagnostics (This element applies to non-surrogate diagnostics.)
<extraResponseData>	<i>structured</i>	Optional, non-repeatable	See Extensions
<echoedSearch RetrieveRequest>	<i>structured</i>	Optional, non-repeatable	see Echoed Request

204
205
206
207
208
209
210
211
212
213
214
215

4.2 Relationship of Actual Elements to Abstract Elements

The following table summarizes the relationship of actual elements to abstract elements defined in the APD. In the first two columns are shown abstract elements and their corresponding actual elements for those abstract elements that have corresponding actual elements in this binding. The third column shows abstract elements for which no corresponding actual elements are defined for this binding. The fourth column lists new elements defined for this binding, that is, for which there are no corresponding abstract elements.

Table 4: Relationship of Actual Elements to Abstract Elements

Abstract Element	Corresponding Actual element	Excluded Abstract Elements	Additional Actual Elements
numberOfItems	numberOfRecords		
resultSetId	resultSetId		
item	record		
nextPosition	nextRecordPosition		
diagnostics	diagnostics		
echoedRequest	echoedSearch RetrieveRequest		
		numberOfGroups	
		nextGroup	
			version
			resultSetIdleTime
			extraResponseData

216 5 Parameter and Element Descriptions

217 All of the parameters and elements are described in this and the following few sections.

- 218 • This section:
 - 219 ○ Describes basic parameters and elements.
- 220 • [Diagnostics](#):
 - 221 ○ Describes the <diagnostics> element.
- 222 • [Extensions](#)
 - 223 ○ describes extension request parameters and response element <extraResponseData>:
- 224 • [Echoed Request](#)
 - 225 ○ describes the element <echoedSearchRetrieveRequest>
- 226 • [Record Serialization and formatting Parameters and Elements](#)
 - 227 ○ Describes the recordPacking, recordSchema, and stylesheet parameters, and the
 - 228 <records> element.

229 5.1 startRecord and maximumRecords

230 The client requests that the server include a range of result set records in the response, beginning with
231 **startRecord** and the number of records supplied is not to exceed **maximumRecords**.

232 startRecord is a positive integer, optional, and its default if omitted is 1. maximumRecords is a non-
233 negative integer, optional, and if omitted, the server may choose any value.

234 The server may return less than the number of records specified by maximumRecords, for example if
235 there are fewer matching records than requested, but MUST NOT return more.

236 5.2 resultSetTTL, and resultSetIdleTime

237 The client may request, via parameter **resultSetTTL**, that the server maintain the result set to be created
238 for a specified period of time (in seconds). The server may choose not to fulfill this request, and may
239 respond with a different value, via the response element resultSetIdleTime.

240 The response element <**resultSetIdleTime**> is a good-faith estimate by the server of the idle time, in
241 seconds. That is, the server projects (but does not guarantee) that the result set will remain available and
242 unchanged (both in content and order) until there is a period of inactivity exceeding this idle time. The idle
243 time must be a positive integer, and should not be so small that a client cannot realistically reference the
244 result set again. If the server does not intend that the result set be referenced, it should omit the result set
245 identifier in the response.

246 <resultSetIdleTime> is independent (from a protocol point of view) of resultSetTTL. It may be less-than,
247 equal-to, or greater-than resultSetTTL, and may be supplied or omitted regardless of whether
248 resultSetTTL is supplied or omitted.

249 5.3 numberOfRecords

250 The server reports the size of the result set via the response element <**numberOfRecords**>. If the query
251 fails, its value MUST be zero.

252 5.4 nextRecordPosition

253 When the last record in the response is not the last result set record, the response may include the
254 element <**nextRecordPosition**> whose value is the ordinal position of the next result set record following
255 the final included record. If there are no remaining records, this element MUST be omitted.

256 5.5 resultSetId

257 The server may supply the identifier of the result set created by the current operation via the response
258 element **<resultSetId>**. Its purpose is to allow the result set to be referenced in a subsequent request..

259 5.6 Echoed Request

260 Very thin clients, such as a web browser with a stylesheet, may not have the facility to record the query
261 that generated the response it has just received. The server may thus echo the request back to the client
262 within the response. There are no request elements associated with this functionality.

263 The response element **<echoedSearchRetrieveRequest>** includes subelements corresponding to
264 request parameters, using the same name.

265 Echoed Request Example

```
266  
267 <echoedSearchRetrieveRequest>  
268   <version>1.2</version>  
269   <query>dc.title = dinosaur</query>  
270   <recordSchema>mods</recordSchema>  
271   <xQuery>  
272     <searchClause xmlns="http://www.loc.gov/zing/cql/xcql/">  
273       <index>dc.title</index>  
274       <relation>  
275         <value>=</value>  
276       </relation>  
277       <term>dinosaur</term>  
278     </searchClause>  
279   </xQuery>  
280   <baseUrl>http://z3950.loc.gov:7090/voyager</baseUrl>  
281 </echoedSearchRetrieveRequest>  
282
```

283 In addition to the echoed parameters, note the sub-elements **<xQuery>** and **<baseUrl>**.

284 **<xQuery>** represents an XCQL rendering of the query. (See XCQL Annex of CQL specification.)

285 Note: This has two benefits.

- 286 • The client can use XSLT or other XML manipulation to modify the query without having a
287 CQL query parser.
- 288 • The server can return extra information specific to the clauses within the query.

289 **<baseUrl>** allows the client to reconstruct queries by simple concatenation, or retrieve the Explain
290 document to fetch additional information such as the title and description to include in the results
291 presented to the user.

292

293 6 Record Serialization and formatting Parameters

294 6.1 recordPacking

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

300 This distinction is made via the request parameter **recordPacking**. If the value of the parameter is 'string',
301 then the server should escape the record before transferring it. If the value is 'xml', then it should embed
302 the XML directly into the response. Either way, the data is transferred within the 'recordData' field. If the
303 server cannot comply with this packing request, then it MUST return a diagnostic.

304 6.2 recordSchema

305 The request Parameter **recordSchema** is the XML schema of the records to be supplied in the response.
306 The value of the parameter is the short name that the server assigns to the identifier for the schema, as
307 listed in the server's Explain file. The default value if not supplied is determined by the server.

308 For example, for the [MODS Schema Version 3.3](http://www.loc.gov/standards/sru/resources/schemas.html) the identifier is info:srw/schema/1/mods-v3.3, as
309 shown in the table at <http://www.loc.gov/standards/sru/resources/schemas.html> and the short name
310 might (but need not) be 'mods'. (Note: schema identifiers are not restricted to those in this table.)

311 The server MUST supply records in the requested schema only. If the schema is unknown or a record
312 cannot be rendered in that schema, then the server MUST return a diagnostic:

- 313 • If the schema is unknown, the server should supply a non-surrogate (fatal) diagnostic:
314 *info:srw/diagnostic/1/66*: "Unknown schema for retrieval".
- 315 • If an individual record cannot be rendered in the requested schema, the server should supply a
316 surrogate (non-fatal) diagnostic in place of the record: *info:srw/diagnostic/1/67*: "Record not
317 available in this schema".

318 6.3 Stylesheet

319 The request parameter **stylesheet** is a URL for a stylesheet. The client requests that the server simply
320 return this URL in the response, in the href attribute of the xml-stylesheet processing instruction before
321 the response xml. (It is likely that the type will be XSL, but not necessarily so.) If the server cannot fulfill
322 this request it must supply a [non-surrogate diagnostic](#) .

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

326 **Example**

```
327 http://z3950.loc.gov:7090/voyager?version=1.2&operation=searchRetrieve  
328 &stylesheet=/master.xsl&query=dinosaur
```

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

```
330 <?xml version="1.0"?>  
331 <?xml-stylesheet type="text/xsl" href="/master.xsl"?>  
332 <sru:searchRetrieveResponse ...
```

333

7 Response Records

334 The response element <records> contains the one or more <record> and surrogate <diagnostic>
335 elements. The <diagnostic> elements occurring within the <records> element represent surrogate
336 diagnostics. These are describes in [Diagnostics](#). Each <record> element is structured into the elements
337 shown in the following table.

338 *Table 5: Structure of the <Record> Element*

Element	Type	Occurrence	Description
<recordSchema>	xs:string	mandatory	The URI identifier of the XML schema in which the record is encoded. Although the request may use the server's assigned short name, the response must always be the full URI.
<recordPacking>	xs:string	mandatory	'string' or 'xml'.
<recordData>	<stringOrXmlFragment>	mandatory	The actual record.
<recordIdentifier>	xs:string	optional	An identifier for the record by which it can unambiguously be retrieved in a subsequent operation. For example via the 'rec.identifier' index in CQL.
<recordPosition>	xs:positiveInteger	optional	The position of the record within the result set.
<extraRecordData>	<xmlFragment>	optional	Any additional information to be transferred with the record.

339 Example

340 An example <records> element with three records:

```

341 <records>
342   <record>
343     <recordSchema>info:srw/schema/1/dc-v1.1</recordSchema>
344     <recordPacking>xml</recordPacking>
345     <recordData>
346       <srw_dc:dc xsi:schemaLocation="info:srw/schema/1/dc-schema
347         http://www.loc.gov/standards/sru/resources/dc-schema.xsd">
348         <title>Fay Vincent Oral History Project collection [videorecording] </title>
349         <creator>Vincent, Fay, interviewer.</creator>
350         <type>Oral histories. aat</type>
351         <language>eng</language>
352         <subject>African American baseball players--Interviews.</subject>
353       </srw_dc:dc>
354     </recordData>
355     <recordPosition>1</recordPosition>
356   </record>
357
358
359
360
361
362   <record>
363     <recordSchema>info:srw/schema/1/dc-v1.1</recordSchema>
364     <recordPacking>xml</recordPacking>

```

```

365     <recordData>
366         <srw_dc:dc xsi:schemaLocation="info:srw/schema/1/dc-schema
367         http://www.loc.gov/standards/sru/resources/dc-schema.xsd">
368             <title>Whitey Ford : a biography /</title>
369             <creator>Coverdale, Miles.</creator>
370             <type>text</type>
371             <publisher>Jefferson, N.C. : McFarland and Co.,</publisher>
372             <date>c2006.</date>
373             <language>eng</language>
374             <description>Includes bibliographical references (p. 233) and index.</description>
375             <subject>Ford, Whitey, 1928-</subject>
376             <identifier>http://www.loc.gov/catdir/toc/ecip0610/2006009578.html</identifier>
377             <identifier>URN:ISBN:0786425148 (pbk. : alk. paper)</identifier>
378         </srw_dc:dc>
379     </recordData>
380     <recordPosition>2</recordPosition>
381 </record>
382
383 <record>
384     <recordSchema>info:srw/schema/1/dc-v1.1</recordSchema>
385     <recordPacking>xml</recordPacking>
386     <recordData>
387         <srw_dc:dc xsi:schemaLocation="info:srw/schema/1/dc-schema
388         http://www.loc.gov/standards/sru/resources/dc-schema.xsd">
389             <title>Whitey Ford sings the blues [sound recording] /</title>
390             <creator>Everlast (Musician) prf</creator>
391             <type>sound recording</type>
392             <publisher>New York, NY : Tommy Boy,</publisher>
393             <date>p1998.</date>
394             <language>eng</language>
395             <description>Rap and rock music.</description>
396             <description>Everlast (vocals, guitars, keyboard, scratches); with assisting musicians </description>
397             <description>"Parental advisory, explicit lyrics"--Container.</description>
398             <description>Compact disc.</description>
399             <description>The white boy is back </description>
400             <subject>Rap (Music)</subject>
401             <subject>Rock music--1991-2000.</subject>
402         </srw_dc:dc>
403     </recordData>
404     <recordPosition>3</recordPosition>
405 </record>
406 </records>
407

```

408 8 Diagnostics

409 Diagnostics are provided in SRU responses both in the response element <diagnostics>, and as a
410 subelement of the response element <records>.

411 A diagnostics is *fatal* or *non-fatal*. Non-fatal diagnostics are further divided into two categories: *surrogate*
412 and *non-surrogate*. See the [diagnostic model](#); to summarize: A surrogate diagnostic replaces a record; a
413 non-surrogate diagnostic refers to the response at large and is supplied in addition and external to the
414 records. A non-surrogate diagnostic may be fatal or non-fatal. So three combinations are possible:

- 415 • surrogate, non-fatal diagnostic (in element <records>)
- 416 • non-surrogate, non-fatal diagnostic (in element <diagnostics>)
- 417 • non-surrogate, fatal diagnostic (in element <diagnostics>)

418 (“Fatal, surrogate” is not a valid combination.)

419 8.1 Diagnostic List

420 See [Diagnostics for use with SRU 1.2](#). This diagnostic list has the namespace: info:srw/diagnostic/1. For
421 example, the URI info:srw/diagnostic/1/10 identifies the diagnostic “Query syntax error”.

422 Diagnostics used in SRU 1.2 need not be limited to this list, nor need this list be used exclusively for
423 SRU 1.2.

424 8.2 Diagnostic Data Elements

425 The [diagnostic schema](#) for SRU 1.2 has three elements, 'uri', 'details' and 'message'.

426 The <uri> element is mandatory and is a URI identifying the particular diagnostic. <details> is optional
427 and contains information specific to the diagnostic. <message>, also optional, contains a human readable
428 message to be displayed.

429 *Table 6: Elements of the Diagnostic Schema*

Element	Type	Occurrence	Description
<uri>	xs:anyURI	Mandatory	The diagnostic's identifying URI.
<details>	xs:string	Optional	Any supplementary information available, often in a format specified by the diagnostic
<message>	xs:string	Optional	A human readable message to display to the end user. The language and style of this message is determined by the server, and clients should not rely on this text being appropriate for all situations.

430

431

432 8.3 Diagnostic Examples

433 These examples are based on the format described above.

434 8.3.1 Non-Surrogate Example

435 **Non-surrogate, fatal diagnostic:**

```
436 <diagnostics>
437   <diagnostic xmlns="info:srw/xmlns/1/sru-1-2-diagnostic">
438     <uri>info:srw/diagnostic/1/38</uri>
439     <details>10</details>
440     <message>Too many boolean operators, the maximum is 10.
441       Please try a less complex query.</message>
442   </diagnostic>
443 </diagnostics>
```

444 8.3.2 Surrogate Example

```
445 Surrogate, non-fatal diagnostic:
446 <records>
447   <record>
448     <recordSchema> info:srw/schema/1/diagnostics-v1.1</recordSchema>
449     <recordData>
450       <diagnostic xmlns=" info:srw/xmlns/1/sru-1-2-diagnostic">
451         <uri>info:srw/diagnostic/1/65</uri>
452         <message>Record deleted by another user.</message>
453       </diagnostic>
454     </recordData>
455   </record> ...
456 </records>
```

457 9 Extensions

458 Both in the request and in the response, additional information may be provided, in the request by an
459 extension parameter (whose name is constructed as described next), and in the response by the
460 <extraResponseData> element.

461 9.1 Extension Request Parameter

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

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

467 Example

```
468 http://z3950.loc.gov:7090/voyager?...&x-info4-onSearchFail=scan
```

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

475 9.2 Extension Response Element: extraResponseData

476 An extension definition may (but need not) define a response, to be carried via the <extraResponseData>
477 element. The extension definition indicates the element names, from the extension's namespace, which
478 will carry the response information.

479 example:

```
480 <sru:extraResponseData>  
481 <auth:token xmlns:auth="info:srw/extension/2/auth-1.0">  
482 277c6d19-3e5d-4f2d-9659-86a77fb2b7c8  
483 </auth:token>  
484 </sru:extraResponseData>
```

485 9.3 Behavior

486 The response may include extraResponseData for a given extension only if the request included the
487 extension parameter for that extension, and the extension definition defines a response. Thus a response
488 may never include unsolicited extraResponseData. For example the response may contain cost
489 information regarding the query or information on the server or database supplying the results. This data
490 must, however, have been requested.

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

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

505 **9.4 Echoing the Extension Request**

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

512 10 Conformance

513 An SRU 1.2 client or server conforms to this standard if it meets the conditions specified in Client
514 Conformance or Server Conformance respectively.

515 10.1 Client Conformance

516 10.1.1 Protocol

517 The client must implement the [protocol model](#). It must support at least one LLP.

518 The SRUP/C must be able to:

- 519 1. Accept a request from the UA.
- 520 2. Assign values to parameters and form Search/Retrieve requests according to the procedures
521 described in the standard.
- 522 3. Compose a PRQ and pass it to the LLP.
- 523 4. Accept a PRS from the LLP.
- 524 5. Decompose the PRS and present information from it to the UA.

525 10.1.2 Query

526 The client must be capable of sending a CQL query. At minimum, level 0 must be supported.

527 10.1.3 Response Format

528 The client must support the SRU 1.2 schema for the response.

529 10.1.4 Diagnostics

530 The client must support the diagnostic schema and be able to present diagnostics received in a PRS to
531 the UA.

532 10.1.5 Explain

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

534 10.2 Server Conformance

535 10.2.1 Protocol

536 The server must implement the protocol model, it must support at least one LLP.

537 The SRUP/S must be able to:

- 538 1. Accept a PRQ from the LLP.
- 539 2. Decompose the PRQ to determine parameter values and interact with the SE as necessary in
540 order to process the request.
- 541 3. Assigning values to elements and compose a PRS according to the procedures described in the
542 standard.
- 543 4. Pass the response to the LLP.

544 10.2.2 Query

545 The server must support CQL queries. At minimum, level 0 must be supported.

546 **10.2.3 Response Format**

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

548 **10.2.4 Diagnostics**

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

551 **10.2.5 Explain**

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

553 **Appendix A. Acknowledgements**

554 Acknowledgements are supplied in the Overview document:

555 *searchRetrieve: Part 0. Overview Version 1.0*

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

557 [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)

558 Appendix B. SRU 1.2 Bindings to Lower Level Protocol

559 Normative Annex

560 B.1 Binding to HTTP GET

561 This annex describes the construction of an SRU 1.2 http: URL to encode parameter values of the form
562 '*key=value*'. Support for Unicode characters is described.

563 B.1.1 Syntax

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

```
566 <base URL>?<searchpart>
```

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

568 Example

569 Assume:

- 570 - The base URL is 'z3950.loc.gov:7090'.
- 571 - The value of parameter 'version' is "1.2".
- 572 - The value of parameter 'operation' is "searchRetrieve".
- 573 - The value of parameter 'query' is "dinosaur".

574

575 Then the URL would be:

```
576 http://z3950.loc.gov:7090/voyager?version=1.2&operation=searchRetrieve  
577 &query=dinosaur
```

578

579 And over the wire goes:

```
580 GET /voyager?version=1.2&operation=searchRetrieve&query=dinosaur HTTP/1.1  
581 Host: z3950.loc.gov:7090
```

582 B.1.2 Encoding (Client Procedure)

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

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

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

593 B.1.3 Decoding (Server Procedure)

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

599 **B.1.4 Example**

600 Consider the following parameter:

601 query=dc.title =/word kirkegård

602 The name of the parameter is "query" and the value is "dc.title =/word kirkegård"

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

606 The following characters must be percent encoded:

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

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

613 query=dc.title%20%3D%2Fword%20kirkeg%C3%A5rd

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

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

616 The Content-type header **MUST** be set to

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

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

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

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

```
624     POST /voyager HTTP/1.1
625     Host: z3850.loc.gov:7090
626     Content-type: application/x-www-form-urlencoded; charset=iso-8859-1
627     Content-length: 51
628     version=1.1&operation=searchRetrieve&query=dinosaur
```

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

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

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

636 **B.3.1 SOAP Requirements**

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

- 638
- SOAP version 1.1 is required. Version 1.2 or higher may be supported.
- 639
- The service style is 'document/literal'.
- 640
- Messages MUST be inline with no multirefs.
- 641
- The SOAPAction HTTP header may be present, but should not be required. If present its value
- 642
- MUST be the empty string. It MUST be expressed as:

643 **SOAPAction: ""**

- 644
- As specified by SOAP, for version 1.1 the Content-type header MUST be 'text/xml'. For version
- 645
- 1.2 the header value MUST be 'application/soap+xml'. (End points supporting both versions of
- 646
- SOAP as well as SRU via POST thus have three content-type headers to consider.)

647 **B.3.2 Parameter Differences**

648 There are some differences regarding the parameters that can be transported via the SOAP binding.

- 649
- The 'operation' request parameter MUST NOT be sent. The operation is determined by the XML
- 650
- constructions employed.
- 651
- The 'stylesheet' request parameter MUST NOT be sent. SOAP prevents the use of stylesheets to
- 652
- render the response.

653 **B.3.3 Example SOAP Request**

```
654     <SOAP:Envelope
655     xmlns:SOAP="http://schemas.xmlsoap.org/soap/envelope/">
656     <SOAP:Body>
657     <SRW:searchRetrieveRequest xmlns:SRW="info:srw/xmlns/1/sru ">
658     <SRW:query>dinosaur</SRW:query>
659     <SRW:startRecord>1</SRW:startRecord>
660     <SRW:maximumRecords>1</SRW:maximumRecords>
661     <SRW:recordSchema>info:srw/schema/1/mods-
662     v3.0</SRW:recordsSchema>
663     </SRW:searchRetrieveRequest>
664     </SOAP:Body>
665     </SOAP:Envelope>
```

666 **B.3.4 WSDL**

667 WSDL for SOAP support can be found at:

668 <http://www.loc.gov/standards/sru/oasis/schemas/sru-wsdl11.wsdl>

669 The SRU request schema associated with the WSDL can be found at:

670 <http://www.loc.gov/standards/sru/oasis/schemas/sruRequest.xsd>

671 (These locations are unofficial. Official locations will be added when known.)

672

673

Appendix C. Diagnostics for use with SRU 1.2

674 Normative Annex

675 The diagnostics below are defined for use with the following SRU namespace: info:srw/diagnostic/1. The
 676 number in the first column identifies the specific diagnostic within that namespace (e.g., diagnostic 2
 677 below is identified by the uri: info:srw/diagnostic/1/2). The “details format” column specifies what should
 678 be returned in the details field. If this column is blank, the format is 'undefined' and the server may return
 679 whatever it feels appropriate, including nothing.

680

General Diagnostics			
Number	Description		Details Format
1	General system error	note	Debugging information (traceback)
2	System temporarily unavailable	note	
3	Authentication error	note	
4	Unsupported operation	note	
5	Unsupported version	note	Highest version supported
6	Unsupported parameter value	note	Name of parameter
7	Mandatory parameter not supplied	note	Name of missing parameter
8	Unsupported Parameter	note	Name of the unsupported parameter
Diagnostics 10-49 reserved for CQL			
Diagnostics Relating to Result Sets			
Number	Description		Details Format
50	Result sets not supported	note	
51	Result set does not exist	note	Result set identifier

52	Result set temporarily unavailable	note	Result set identifier
53	Result sets only supported for retrieval	note	
54	Not used.		
55	Combination of result sets with search terms not supported	note	
56	Not used.		
57	Not used.		
58	Result set created with unpredictable partial results available	note	
59	Result set created with valid partial results available	note	
60	Result set not created: too many matching records	note	Maximum number
Diagnostics Relating to Records			
Number	Description		Details Format
61	First record position out of range	note	
62	Not used.		
63	Not used.		
64	Record temporarily unavailable	note	
65	Record does not exist	note	
66	Unknown schema for retrieval	note	Schema URI or short name
67	Record not available in this schema	note	Schema URI or short name
68	Not authorized to send record	note	
69	Not authorized to send record in this schema	note	
70	Record too large to send	note	Maximum record size
71	Unsupported record packing	note	

72	XPath retrieval unsupported	note	
73	XPath expression contains unsupported feature	note	Feature
74	Unable to evaluate XPath expression	note	
Diagnostics Relating to Explain			
Number	Description	Details Format	
100	Not used.		
101	Not used.		
102	Not used.		
Diagnostics relating to Stylesheets			
Number	Description	Details Format	
110	Stylesheets not supported	note	
111	Unsupported stylesheet	note	URL of stylesheet
Diagnostics 120-121 reserved for Scan			

C.1 Notes

No.	Cat.	Description	Notes/Examples
1	general	General system error	The server returns this error when it is unable to supply a more specific diagnostic. The sever may also optionally supply debugging information.
2	general	System temporarily unavailable	The server cannot respond right now, perhaps because it's in a maintenance cycle, but will be able to in the future.
3	general	Authentication error	The request could not be processed due to lack of authentication.
4	general	Unsupported operation	Currently three operations are defined -- searchRetrieve, explain, and scan. searchRetrieve and explain are mandatory, so this diagnostic would apply only to scan, or in SRU where an undefined operation is sent.
5	general	Unsupported version	Currently only version 1.1 is defined and so this diagnostic has no meaning. In the future, when another version is defined, for example version 1.2, this diagnostic may be returned when the server receives a request where the version parameter indicates 1.2, and the server doesn't support version 1.2.

6	general	Unsupported parameter value	This diagnostic might be returned for a searchRetrieve request which includes the recordPacking parameter with a value of 'xml', when the server does not support that value. The diagnostic might supply the name of parameter, in this case 'recordPacking'.
7	general	Mandatory parameter not supplied	This diagnostic might be returned for a searchRetrieve request which omits the query parameter. The diagnostic might supply the name of missing parameter, in this case 'query'.
8	general	Unsupported Parameter	This diagnostic might be returned for a searchRetrieve request which includes the recordXPath parameter when the server does not support that parameter. The diagnostic might supply the name of unsupported parameter, in this case 'recordXPath'.
50	result set	Result sets not supported	The server cannot create a persistent result set.
51	result set	Result set does not exist	The client asked for a result set in the query which does not exist, either because it never did or because it had expired.
52	result set	Result set temporarily unavailable	The result set exists, it cannot be accessed, but will be able to be accessed again in the future.
53	result set	Result sets only supported for retrieval	Other operations on results apart from retrieval, such as sorting them or combining them, are not supported.
55	result set	Combination of result sets with search terms not supported	Existing result sets cannot be combined with new terms to create new result sets. eg cql.resultsetid = foo not dc.title any fish
58	result set	Result set created with unpredictable partial results available	The result set is not complete, possibly due to the processing being interrupted mid way through. Some of the results may not even be matches.
59	result set	Result set created with valid partial results available	All of the records in the result set are matches, but not all records that should be there are.
60	result set	Result set not created: too many matching records	There were too many records to create a persistent result set.
61	records	First record position out of range	For example, if the request matches 10 records, but the start position is greater than 10.
64	records	Record temporarily unavailable	The record requested cannot be accessed currently, but will be able to be in the future.
65	records	Record does not exist	The record does not exist, either because it never did, or because it has subsequently been deleted.
66	records	Unknown schema for retrieval	The record schema requested is unknown. Eg. the client asked for MODS when the server can only return simple Dublin Core
67	records	Record not available in	The record schema is known, but this particular record cannot be

		this schema	transformed into it.
68	records	Not authorized to send record	This particular record requires additional authorisation in order to receive it.
69	records	Not authorized to send record in this schema	The record can be retrieved in other schemas, but the one requested requires further authorization.
70	records	Record too large to send	The record is too large to send.
71	records	Unsupported record packing	The server supports only one of string or xml, or the client requested a recordPacking which is unknown.
72	records	XPath retrieval unsupported	The server does not support the retrieval of nodes from within the record.
73	records	XPath expression contains unsupported feature	Some aspect of the XPath expression is unsupported. For example, the server might be able to process element nodes, but not functions.
74	records	Unable to evaluate XPath expression	The server could not evaluate the expression, either because it was invalid or it lacks some capability.
110	stylesheet	Stylesheets not supported	The SRU server does not support stylesheets, or a stylesheet was requested from an SRW server.
111	stylesheet	Unsupported stylesheet	This particular stylesheet is not supported, but others may be.

681