searchRetrieve: Part 2. searchRetrieve Operation: APD Binding for SRU 1.2 Version 1.0

OASIS Standard

30 January 2013

Specification URIs

This version:

http://docs.oasis-open.org/search-ws/searchRetrieve/v1.0/os/part2-sru1.2/searchRetrieve-v1.0-os-part2-sru1.2.doc (Authoritative)
http://docs.oasis-open.org/search-ws/searchRetrieve/v1.0/os/part2-sru1.2/searchRetrieve-v1.0-os-part2-sru1.2.html
http://docs.oasis-open.org/search-ws/searchRetrieve/v1.0/os/part2-sru1.2/searchRetrieve-v1.0-os-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 Dixson (ldix@loc.gov), Library of Congress
Ralph Levan (levan@oclc.org), OCLC
Janifer Gatenby (Janifer.Gatenby@oclc.org), OCLC
Tony Hammond (t.hammond@nature.com), Nature Publishing Group
Matthew Dovey (m.dovey@jisc.ac.uk), JISC Executive, University of Bristol

Additional artifacts:

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

- XML schemas: http://docs.oasis-open.org/search-ws/searchRetrieve/v1.0/os/schemas/
- *searchRetrieve: Part 0. Overview Version 1.0*. http://docs.oasis-open.org/search-ws/searchRetrieve/v1.0/os/part0-overview/searchRetrieve-v1.0-os-part0-overview.html
- *searchRetrieve: Part 1. Abstract Protocol Definition Version 1.0*. http://docs.oasis-open.org/search-ws/searchRetrieve/v1.0/os/part1-apd/searchRetrieve-v1.0-os-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/os/part2-sru1.2/searchRetrieve-v1.0-os-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/os/part3-sru2.0/searchRetrieve-v1.0-os-part3-sru2.0.html
- *searchRetrieve: Part 4. APD Binding for OpenSearch Version 1.0*. http://docs.oasis-open.org/search-ws/searchRetrieve/v1.0/os/part4-opensearch/searchRetrieve-v1.0-os-part4-opensearch.html
- *searchRetrieve: Part 5. CQL: The Contextual Query Language Version 1.0*. http://docs.oasis-open.org/search-ws/searchRetrieve/v1.0/os/part5-cql/searchRetrieve-v1.0-os-part5-cql.html
- *searchRetrieve: Part 6. SRU Scan Operation Version 1.0*. http://docs.oasis-open.org/search-ws/searchRetrieve/v1.0/os/part6-scan/searchRetrieve-v1.0-os-part6-scan.html
- *searchRetrieve: Part 7. SRU Explain Operation Version 1.0*. http://docs.oasis-open.org/search-ws/searchRetrieve/v1.0/os/part7-explain/searchRetrieve-v1.0-os-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 membership of OASIS 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. 30 January 2013. OASIS Standard. http://docs.oasis-open.org/search-ws/searchRetrieve/v1.0/os/part2-sru1.2/searchRetrieve-v1.0-os-part2-sru1.2.html.

Notices

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

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

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

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

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

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

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

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

Table of Contents

1 Introduction

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

This document, "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.

2 Overview and Model

2.1 Relationship to Abstract Protocol Definition

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

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

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

2.2 Operation Model

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

For the SRU protocol, three operations are defined:

1. **SearchResponse Operation**. The SearchResponse operation is defined by the SRU protocol, which is this specification.
2. **Scan Operation**.
3. **Explain Operation**. See Explain Model below.

2.3 Data model

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

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

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

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

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.

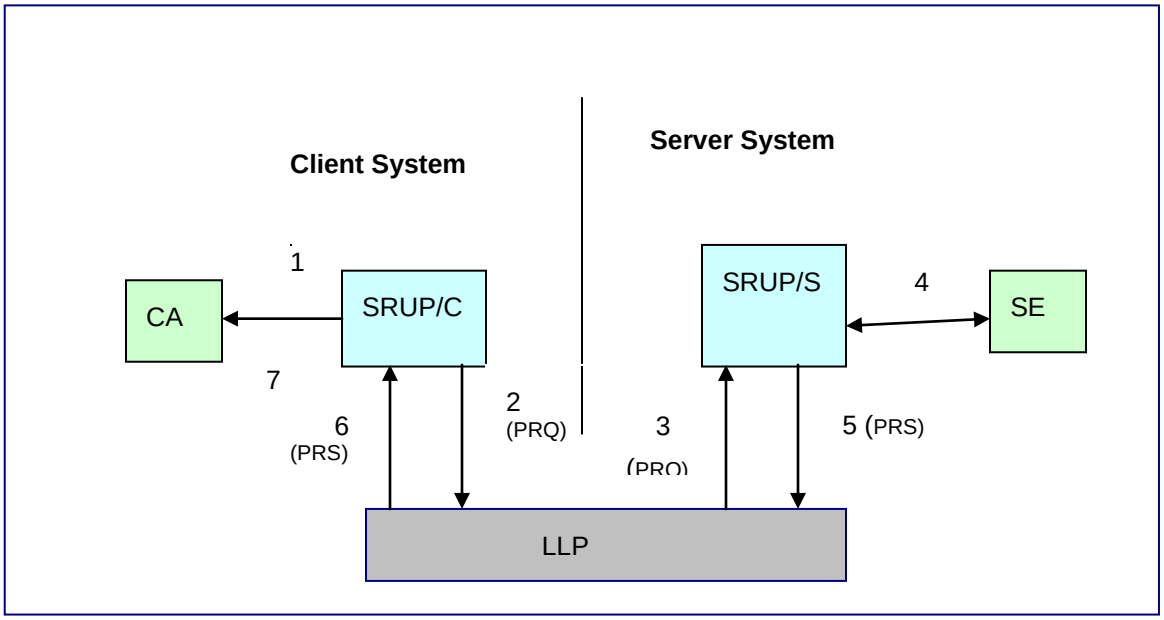
2.4 Protocol Model

The protocol model assumes these conceptual components:

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

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

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



The protocol model is described diagrammatically in this picture.

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

3. LLP passes the PRQ to SRUP/S.
4. SRUP/S interacts with SE to form a PRS.
5. The PRS is passed to the LLP.
6. LLP passes the PRS to SRUP/C.
7. SRUP/C presents results to CA

Processing Model

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

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

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

2.5 Query model

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

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

2.6 Result Set Model

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

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

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

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

Thus, suppose an abstract record is deleted or otherwise becomes unavailable while a result set which references that record still exists. This MUST not cause re-ordering. For example, if a client retrieves records 1 through 3, and subsequently record 2 becomes unavailable, if the server again requests record 3, it must be the same record 3 that was returned as record 3 in the earlier operation. ("Same record" does not necessarily mean the same content; the record's content may have changed.) If the server requests record 2 (no longer available) the server should supply a surrogate diagnostic (see Diagnostic 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.

2.7 Diagnostic Model

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

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

2.8 Explain Model

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

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

3 Request Parameters (Summary)

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

3.1 Actual Request Parameters for this Binding

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

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

3.2 Relationship of Actual Parameters to Abstract Parameters

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

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		
responseItemType	recordSchema		

		SortOrder	
		responseFormat	
		Group	
			operation
			version
			recordPacking
			resultSetTTL
			stylesheet
			Extension parameters

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.

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>	*xs:string*	Mandatory, non-repeatable	` The string '1.2'.
<numberOfRecords>	*xs:integer*	Mandatory, non-repeatable	See numberOfRecords
<resultSetId>	*xs:string*	Optional, non-repeatable	See resultSetId
<resultSetIdleTime>	*xs:integer*	Optional, non-repeatable	see resultSetTTL and resultSetIdleTime.
<records>	*structured*	Optional, repeatable	see records
<nextRecordPosition>	*xs:integer*	Optional, non-repeatable	see nextRecordPosition
<diagnostics>	*structured*	Optional, non-repeatable	see diagnostics (This element applies to non-surrogate diagnostics.)
<extraResponseData>	*structured*	Optional, non-repeatable	See Extensions
<echoedSearch RetrieveRequest>	*structured*	Optional, non-repeatable	see *Echoed Request*

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

5 Parameter and Element Descriptions

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

- This section**:**
  - Describes basic parameters and elements.
- Diagnostics:
  - Describes the <diagnostics> element.
- Extensions
  - describes extension request parameters and response element <extraResponseData>:
- Echoed Request
  - describes the element <echoedSearchRetrieveRequest>
- Record Serialization and formatting Parameters and Elements
  - Describes the recordPacking, recordSchema, and stylesheet parameters, and the <records> element.

5.1 startRecord and maximumRecords

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

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

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

5.2 resultSetTTL, and resultSetIdleTime

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

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

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

5.3 numberofRecords

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

5.4 nextRecordPosition

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

5.5 resultSetId

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

5.6 Echoed Request

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

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

Echoed Request Example

```
<echoedSearchRetrieveRequest>
        <version>1.2</version>
        <query>dc.title = dinosaur</query>
        <recordSchema>mods</recordSchema>
       <xQuery>
             <searchClause xmlns="http://www.loc.gov/zing/cql/xcql/">
                   <index>dc.title</index>
                   <relation>
                        <value>=</value>
                  </relation>
                  <term>dinosaur</term>
             </searchClause>
       </xQuery>
      <baseUrl>http://z3950.loc.gov:7090/voyager</baseUrl>
</echoedSearchRetrieveRequest>
```

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

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

Note: This has two benefits.

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

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

6 Record Serialization and formatting Parameters

6.1 recordPacking

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

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

6.2 recordSchema

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

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

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

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

6.3 Stylesheet

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

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

Example

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

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

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

7 Response Records

The response element <**records**> contains the one or more <record> and surrogate <diagnostic> elements. The <diagnostic> elements occurring within the <records> element represent surrogate diagnostics. These are describes in Diagnostics. Each <record> element is structured into the elements shown in the following table.

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.

Example

An example <records> element with three records:

```
<records>
        <record>
                <recordSchema>info:srw/schema/1/dc-v1.1</recordSchema>
                <recordPacking>xml</recordPacking>
                <recordData>
                        <srw_dc:dc xsi:schemaLocation="info:srw/schema/1/dc-schema
                            http://www.loc.gov/standards/sru/resources/dc-schema.xsd">
                                <title>Fay Vincent Oral History Project collection [videorecording]  /</title>
                                <creator>Vincent, Fay, interviewer.</creator>
                                <type>Oral histories. aat</type>
                                <language>eng</language>
                                <subject>African American baseball players--Interviews.</subject>
                        </srw_dc:dc>
                </recordData>
                <recordPosition>1</recordPosition>
        </record>
        <record>
                <recordSchema>info:srw/schema/1/dc-v1.1</recordSchema>
                <recordPacking>xml</recordPacking>
```



```
                    <recordData>
                        <srw_dc:dc xsi:schemaLocation="info:srw/schema/1/dc-schema
                        http://www.loc.gov/standards/sru/resources/dc-schema.xsd">
                            <title>Whitey Ford : a biography /</title>
                            <creator>Coverdale, Miles.</creator>
                            <type>text</type>
                            <publisher>Jefferson, N.C. : McFarland and  Co.,</publisher>
                            <date>c2006.</date>
                            <language>eng</language>
                            <description>Includes bibliographical references (p. 233) and index.</description>
                            <subject>Ford, Whitey, 1928-</subject>
                            <identifier>http://www.loc.gov/catdir/toc/ecip0610/2006009578.html</identifier>
                            <identifier>URN:ISBN:0786425148 (pbk. : alk. paper)</identifier>
                        </srw_dc:dc>
                    </recordData>
                    <recordPosition>2</recordPosition>
                </record>

<record>
                    <recordSchema>info:srw/schema/1/dc-v1.1</recordSchema>
                    <recordPacking>xml</recordPacking>
                    <recordData>
                        <srw_dc:dc xsi:schemaLocation="info:srw/schema/1/dc-schema
                         http://www.loc.gov/standards/sru/resources/dc-schema.xsd">
                            <title>Whitey Ford sings the blues [sound recording] /</title>
                            <creator>Everlast (Musician) prf</creator>
                            <type>sound recording</type>
                            <publisher>New York, NY : Tommy Boy,</publisher>
                            <date>p1998.</date>
                            <language>eng</language>
                            <description>Rap and rock music.</description>
                            <description>Everlast (vocals, guitars, keyboard, scratches); with assisting musicians </description>
                            <description>"Parental advisory, explicit lyrics"--Container.</description>
                            <description>Compact disc.</description>
                            <description>The white boy is back </description>
                            <subject>Rap (Music)</subject>
                            <subject>Rock music--1991-2000.</subject>
                        </srw_dc:dc>
                    </recordData>
                    <recordPosition>3</recordPosition>
                </record>
</records>
```


8 Diagnostics

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

A diagnostics is *fatal*' or *non-fatal. N*on-fatal diagnostics are further divided into two categories: *surrogate* and *non-surrogate.* See the diagnostic model; to summarize: A surrogate diagnostic replaces a record; a non-surrogate diagnostic refers to the response at large and is supplied in addition and external to the records. A non-surrogate diagnostic may be fatal or non-fatal. So three combinations are possible:

- surrogate, non-fatal diagnostic (in element <records>)
- non-surrogate, non-fatal diagnostic (in element <diagnosics>)
- non-surrogate, fatal diagnostic (in element <diagnosics>)

("Fatal, surrogate" is not a valid combination.)

8.1 Diagnostic List

See Diagnostics for use with SRU 1.2. This diagnostic list has the namespace: info:srw/diagnostic/1. For example, the URI info:srw/diagnostic/1/10 identifies the diagnostic "Query syntax error".

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

8.2 Diagnostic Data Elements

The diagnostic schema for SRU 1.2 has three elements, 'uri', 'details' and 'message'.

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

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.

8.3 Diagnostic Examples

These examples are based on the format described above.

8.3.1 Non-Surrogate Example

```
Non-surrogate, fatal diagnostic:
```



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

8.3.2 Surrogate Example

```
Surrogate, non-fatal diagnostic:
<records>
      <record>
         <recordSchema> info:srw/schema/1/diagnostics-v1.1</recordSchema>
         <recordData>
              <diagnostic xmlns=" info:srw/xmlns/1/sru-1-2-diagnostic">
                     <uri>info:srw/diagnostic/1/65</uri>
                     <message>Record deleted by another user.</message>
              </diagnostic>
         </recordData>
     </record> ...
<records>
```


9 Extensions

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

9.1 Extension Request Parameter

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

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

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

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

9.2 Extension Response Element: extraResponseData

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

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

9.3 Behavior

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

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

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

9.4 Echoing the Extension Request

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

10 Conformance

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

10.1 Client Conformance

10.1.1 Protocol

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

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

10.1.2 Query

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

10.1.3 Response Format

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

10.1.4 Diagnostics

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

10.1.5 Explain

The client must be able to retrieve the Explain record.

10.2 Server Conformance

10.2.1 Protocol

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

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

10.2.2 Query

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

10.2.3 Response Format

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

10.2.4 Diagnostics

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

10.2.5 Explain

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

Appendix A. Acknowledgements

Acknowlegements are supplied in the Overview document:
searchRetrieve: Part 0. Overview Version 1.0
http://docs.oasis-open.org/search-ws/searchRetrieve/v1.0/csd01/part0-overview/searchRetrieve-v1.0-csd01-part0-overview.doc

Appendix B. SRU 1.2 Bindings to Lower Level Protocol

Normative Annex

B.1 Binding to HTTP GET

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

B.1.1 Syntax

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

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

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

Example

```
Assume:
 -   The base URL is 'z3950.loc.gov:7090'.
 -   The value of parameter 'version' is "1.2".
 -   The value of parameter 'operation' is '"searchRetrieve".
 -   The value of parameter 'query' is "dinosaur".

Then the URL would be:
http://z3950.loc.gov:7090/voyager?version=1.2&operation=searchRetrieve
&query=dinosaur

And over the wire goes:
GET /voyager?version=1.2&operation=searchRetrieve&query=dinosaur HTTP/1.1
Host: z3950.loc.gov:7090
```

B.1.2 Encoding (Client Procedure)

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

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

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

B.1.3 Decoding (Server Procedure)

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

B.1.4 Example

Consider the following parameter:

query=dc.title =/word kirkegård

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

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

The following characters must be percent encoded:

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

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

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

B.2 Binding to HTTP POST

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

The Content-type header MUST be set to

application/x-www-form-urlencoded'

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

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

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

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

B.3 Binding to HTTP SOAP

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

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

B.3.1 SOAP Requirements

The specification adheres to the Web Services Interoperability recommendations.

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

SOAPAction: ""

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

B.3.2 Parameter Differences

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

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

B.3.3 Example SOAP Request

```
<SOAP:Envelope
xmlns:SOAP="http://schemas.xmlsoap.org/soap/envelope/">
  <SOAP:Body>
   <SRW:searchRetrieveRequest xmlns:SRW="info:srw/xmlns/1/sru ">
    <SRW:query>dinosaur</SRW:query>
    <SRW:startRecord>1</SRW:startRecord>
    <SRW:maximumRecords>1</SRW:maximumRecords>
    <SRW:recordSchema>info:srw/schema/1/mods-
v3.0</SRW:recordsSchema>
   </SRW:searchRetrieveRequest>
  </SOAP:Body>
</SOAP:Envelope>
```

B.3.4 WSDL

WSDL for SOAP support can be found at:

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

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

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

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

Appendix C. Diagnostics for use with SRU 1.2

Normative Annex

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

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 interupted 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 futher 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

