Search Web Services Version 1.0

Discussion Document

2 November 2007

URIs:

http://docs.oasis-open.org/search-ws/v1.0/DiscussionDocument.doc
http://docs.oasis-open.org/search-ws/v1.0/DiscussionDocument.pdf
http://docs.oasis-open.org/search-ws/v1.0/DiscussionDocument.html

Technical Committee:

OASIS Search Web Services TC

Chair(s):

Ray Denenberg

Matthew Dovey

Related work:

This specification replaces or supercedes:

- SRU 1.2

This specification is related to:

- ISO 23950
- NISO Z39.92

Status:

This document has no official status. It was prepared by the OASIS Search Web Services TC as a strawman proposal, for public review, intended to generate discussion. It is not a Committee Draft.

Purpose of this Document

This specification is based on the SRU (Search Retrieve via URL) specification which can be found at http://www.loc.gov/standards/sru/. It is expected that this standard, when published, will deviate from SRU. How much it will deviate cannot be predicted at this time. The fact that the SRU spec is used as a starting point for development should not be cause for concern that this might be an effort to fast track SRU. The committee hopes to preserve the useful features of SRU, but not to preserve those that are not considered useful.

The OASIS Technical Committee developing this standard has decided to request OASIS to release this as a discussion document. Detailed review of this document is premature at this point, but feedback on the functionality and approach is solicited.

Copyright © OASIS® 1993–2007. All Rights Reserved.

Open Issues

There are several current open issues before the committee not reflected in the body of the document.

There is a wiki for the committee at http://wiki.oasis-open.org/search-ws/FrontPage, and an issues list at http://wiki.oasis-open.org/search-ws/issues

These issues are summarized here:

1. **Binary representation within records**
   The protocol must support the inclusion of binary objects within records. And external mechanisms exist to provide this support. The issue is whether the standard needs to define an explicit mechanism.

2. **Parameterized query support**
   The protocol should support parameterized queries. Should they be supported within CQL, should CQL be a special case of parameterized query, or should these two be defined separately.

3. **OpenSearch**
   The specification is intended to subsume the OpenSearch functionality. The existing OpenSearch specification is regarded as a legacy specification and this standard will also and show how the protocol interoperates with that spec. This has not been sufficiently addressed in this draft.

4. **XML/WSDL**
   The committee determined that it is premature to write XML/WSDL for the protocol, so there is a stub section with a pointer to the current SRU xml. XML/WSDL will be written later.

5. **Operation Parameter**
   There is a suggestion to eliminate the operation parameter, incorporating it instead in the base url, in some fashion. (This is not done in this draft.) The reason for the suggestion is that this parameter is not consistent with REST principles.

6. **ATOM (or RSS) as a response schema.**
   There is a proposal to replace the SRU response schema with ATOM or RSS. The current draft adds a parameter allowing the client to request an alternative schema. There should be one schema singled out in the standard that is mandatory. Currently that would be the SRU response schema, and the proposal is to make ATOM or RSS the single required schema instead.

7. **Scan**
   There is a suggestion to eliminate the Scan operation, and instead represent this functionality via search/retrieve.

8. **XCQL**
   There is a suggestion is to eliminate XCQL, which is an XML representation of the CQL query - it is not used in a request, only in the echoed response. Some impementors find it useful to have the query echoed in a parsed form. However its existence causes confusion.

9. **State**
   There is discussion within the committee over how stateful the protocol (as currently defined) is. Some say it is not stateful at all. Others feel that the result set model is stateful. Actually there are two points of debate: whether the protocol is stateful, and whether it should be.

Copyright © OASIS® 1993–2007. All Rights Reserved.

Notices

Copyright © OASIS® 2007. 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 names "OASIS", [insert specific trademarked names, abbreviations, etc. here] are trademarks of OASIS, the owner and developer of this specification, and should be used only to refer to the organization and its official outputs. OASIS welcomes reference to, and implementation and use of, specifications, while reserving the right to enforce its marks against misleading uses. Please see http://www.oasis-open.org/who/trademark.php for above guidance.

Table of Contents

1 Introduction 7
1.1 Terminology 7
1.2 Normative References 7
1.3 Non-Normative References 7
2 Search Web Service Overview 8
3 Contextual Query Language 9
3.1 Query Syntax 9
3.1.1 Basic Query Structure 9
3.1.2 Search Clause 9
3.1.3 Search Term 9
3.1.4 Index Name 10
3.1.5 Relation 10
3.1.6 Relation Modifiers 10
3.1.7 Boolean Operators 11
3.1.8 Boolean Modifiers 11
3.1.9 Proximity Modifiers 12
3.1.10 Sorting 12
3.1.11 Prefix Assignment 13
3.1.12 Case Sensitivity 13
3.2 BNF 13
3.3 Context Sets 14
4 The searchRetrieve operation 16
4.1 Request Parameters 16
4.2 Response Parameters 17
4.3 Version: the “version” Parameter 18
4.4 Records 18
4.4.1 Record Parameters 18
4.4.2 Record Packing 19
4.5 Result Sets 20
4.5.1 Result Set Model 20
4.5.2 resultSetId 20
4.5.3 ResultSet Idle Time 21
4.6 Diagnostics 21
4.6.1 Diagnostic Categories: Fatal vs. Non-fatal, and Surrogate Vs. Non-Surrogate 21
4.6.2 Diagnostic Schema 21
4.7 Extensions: the “extraRequestData’, ‘extraResponseData’, and xtraRecordData’ Parameters 23
4.8 Echoing the Request: The “echoedSearchRetrieveRequest” Parameter 24
4.8.1 xQuery 24
4.8.2 baseUrl 24
4.9 Stylesheets: the ‘stylesheet’ Parameter 25
5 Scan Operation 26
5.1 Request Parameters 26
5.2 Response Parameters 27

Copyright © OASIS® 1993–2007. All Rights Reserved.

5.3 Terms 27
5.4 Example Scan Response 28
6 The Explain Facility 30
6.1 Explain Operation 30
6.1.1 Request Parameters 30
7 XML and WSDL Files 31
8 Transports 32
8.1 HTTP Get Binding 32
8.1.1 Syntax 32
8.1.2 Encoding Issues 32
8.1.3 Server Procedure 33
8.2 HTTP Post Binding 33
8.3 SOAP Binding 34
8.3.1 SOAP Requirements 34
8.3.2 SOAP Parameter Differences 34
8.3.3 Extension Parameters via SOAP 35
A. The CQL Context Set 36
A.1 Indexes 36
A.2 *Relations* 37
A.2.1 Implicit Relations 37
A.2.2 Defined Relations 38
A.3 Relation Modifiers 39
A.3.1 Functional Modifiers 39
A.3.2 Term-format Modifiers 40
A.3.3 Masking 41
A.4 Booleans 43
A.5 Boolean Modifiers 43
Note about Proximity Units 44
B. Diagnostics 45
C. NISO Z39.92 (ZeeRex) 58
D. OpenSearch 60
D.1 OpenSearch Description Document 60
D.2 OpenSearch URL Template 61
D.3 OpenSearch Response Elements 61
E. Authentication, Authorization, and Access Control 63
E.1 Authentication 63
E.2 Authorization and Access Control 63
E.3 IP Address 63
Users may be differentiated by the IP address from which they are connecting to the server. Unfortunately this is unreliable at best due to the increasing use of web proxy systems -- there may be many users all of which appear to be coming from the same IP address due to a proxy. The advantage is that it is completely transparent to the client and hence the user, so for a small service may be appropriate. 63
E.4 Basic Authentication 63
E.5 Secure Sockets 64
E.6 Additional Message Data 64

Copyright © OASIS® 1993–2007. All Rights Reserved.

E.7 Web Services Security and Security Assertion Markup Language (SAML) Security Tokens 64

Copyright © OASIS® 1993–2007. All Rights Reserved.

1 Introduction

[All text is normative unless otherwise labeled]

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 Normative References

[RFC2119] S. Bradner, *Key words for use in RFCs to Indicate Requirement Levels*, http://www.ietf.org/rfc/rfc2119.txt, IETF RFC 2119, March 1997.

[Reference] [Full reference citation]

1.3 Non-Normative References

[Reference] [Full reference citation]

Copyright © OASIS® 1993–2007. All Rights Reserved.

2 Search Web Service Overview

The Search web service is a means of opening a database to external enquiry in a standardized manner that facilitates discovery of query and response possibilities and makes it possible for heterogeneous databases to be queried simultaneously with the same or similar queries. Client software can be easily configured using a standardized XML explain document that is accessible from the base URL or via the explain operation. In contrast with protocols such as SQL and XQuery, detailed knowledge of a database's structure is not necessary as the explain document contains parsable information on server defaults, searchable indexes and record schemas that are returned in the response.

Context sets can be made for use with the search web service that define standard index names and search attributes thus facilitating multi-database searching via either a single or similar searches. Profiles can be registered combining context sets and record schemas and so ensure inter-operability in a variety of domains.

Two kinds of enquiry access are defined; search via keywords or phrases that returns a result set of records and scan via terms that returns a list of terms in an index.

A search or scan can be expressed in a simple URL, enabling a search to be embedded in any web page. The server may send the results with an accompanying XML style sheet, thus the service can be widely used in web pages without any underlying programming.

Copyright © OASIS® 1993–2007. All Rights Reserved.

3 Contextual Query Language

CQL, the *Contextual Query Language*, is a formal language for representing queries to information retrieval systems such as web indexes, bibliographic catalogs and museum collection information. The design objective is that queries be human readable and writable, and that the language be intuitive while maintaining the expressiveness of more complex languages.

Traditionally, query languages have fallen into two camps: Powerful, expressive languages, not easily readable nor writable by non-experts (e.g. SQL, PQF, and XQuery);or simple and intuitive languages not powerful enough to express complex concepts (e.g. CCL and google). CQL tries to combine simplicity and intuitiveness of expression for simple, every day queries, with the richness of more expressive languages to accommodate complex concepts when necessary.

3.1 Query Syntax

3.1.1 Basic Query Structure

A CQL query consists of either a single search clause [example a], or multiple search clauses connected by boolean operators [example b]. It may have a sort specification at the end, following the 'sortBy' keyword [example c]. In addition it may include prefix assignments which assign short names to context set identifiers [example d].

Examples:

a. dc.title = fish

b. dc.title = fish or dc.creator = sanderson

c. dc.title = fish sortBy dc.date/sort.ascending

d. > dc = "info:srw/context-sets/1/dc-v1.1" dc.title any fish

3.1.2 Search Clause

A search clause consists of either an index, relation and a search term [example a], or a search term by itself [example b]. If the clause consists of just a term, then the index is treated as 'cql.serverChoice', and the relation is treated as '=' [example c]. (Therefore example b and c are semantically equivalent.)

Examples:

a. dc.title = fish

b. fish

c. cql.serverChoice = fish

3.1.3 Search Term

Search terms MAY be enclosed in double quotes [example a], though need not be [example b]. Search terms MUST be enclosed in double quotes if they contain any of the following characters: < > = / ( ) and whitespace [example c]. The search term may be an empty string [example d], but must be present in a search clause. The empty search term has no defined semantics.

Copyright © OASIS® 1993–2007. All Rights Reserved.

Examples:

a. "fish"

b. fish

c. "squirrels fish"

d. “”

3.1.4 Index Name

An index name always includes a base name [example a] and may also include a prefix [example b], which determines the context set of which the index is a part. The base name and the prefix are separated by a dot character ('.'). If multiple '.' characters are present, then the first should be treated as the prefix/base name delimiter. If the prefix is not supplied, it is determined by the server. Examples:

Examples:

a. title any Afish dog@

b. dc.title any Afish dog@

3.1.5 Relation

The relation in a search clause specifies the relationship between the index and search term. It also always includes a base name [example a] and may also include a prefix providing a context for the relation [example b]. If a relation does not have a prefix, the context set is 'cql'. If no relation is supplied in a search clause, then = is assumed, which means that the relation is determined by the server. (As is noted above, if the relation is omitted then the index MUST also be omitted; the relation is assumed to be A=@ and the index is assumed to be cql.serverChoice; that is, the server choses both the index and the relation.)

Examples:

a. dc.title any “fish frog”
Find records where the title (as defined by the Adc@ context set) contains one of the words :fish@, Afrog@

b. dc.title cql.any “fish frog”
This query has the same meaning as the previous, since the default context set for the relation is Acql@.

c. dc.title cql.all “fish frog”
Find records where the title contains all of the words :fish@, Afrog@

3.1.6 Relation Modifiers

Relations may be modified by one or more relation modifiers. Relation modifiers always include a base name, and may include a prefix for a context set [example a] as above. If a prefix is not supplied, the context set is 'cql'. Relation modifiers are separated from each other and from the relation by forward slash characters('/'). Whitespace may be present on either side of a '/' character, but the relation plus modifiers group may not end in a '/' [example b]. Relation modifiers may also have a comparison symbol and a value. The comparison symbol is any of = < <= > >= <>. The value must obey the same rules for quoting as search terms, above [example c].

Examples:

Copyright © OASIS® 1993–2007. All Rights Reserved.

a. dc.title any/relevant fish
The relation modifier Arelevant@ means The server should use a relevancy algorithm for determining matches and the order of the result set. When the relevant modifier is used, the actual relation is often not significant.

b. dc.title any/ relevant /cql.string fish

(we need to explain this one or drop it.)

c. title any/rel.algorithm=cori fish
This example is distinguished from example 1 in which the modifier Arelevant@ is from the CQL context set. In this case the modifier is Aalgorithm=core@, from the rel context set, in essence meaning use the relevance algorithm Acori@. A description of this context set is available at http://srw.cheshire3.org/contextSets/rel/

3.1.7 Boolean Operators

Search clauses may be linked by boolean operators. These are: **and**, **or**, **not** and **prox** [example in 3.1.8]. Note that **not** is 'and-not' and must not be used as a unary operator. Boolean operators all have the same precedence; they are evaluated left-to-right. Parentheses may be used to override left-to-right evaluation [example b].

Examples:

a. dc.title = “monkey house” and dc.creator = vonnegut

b. dc.title = “monkey house” not dc.creator = vonnegut

c. dc.title = fish or dc.creator = sanderson

d. dc.title = fish or (dc.creator = sanderson and dc.identifier = "id:1234567")

3.1.8 Boolean Modifiers

Booleans may be modified by one or more boolean modifiers, separated as per relation modifiers with '/' characters. Again, boolean modifiers consist of a base name and may include a prefix determining the modifier's context set [example a]. If not supplied, then the context set is 'cql'. As per relation modifiers, they may also have a comparison symbol and a value [example b].

Examples:

a. dc.title = fish or/rel.combine=sum dc.creator any sanderson

[We need an explanation here of what relevance means when applied to a boolean (as opposed to a relation). We never have understood this. If we can=t describe it then delete this example.]

b. dc.title = monkey prox/unit=word/distance>1 dc.title = house
Find records where both Amonkey@ and Ahouse@ are in the title, separated by at least one intervening word.

Copyright © OASIS® 1993–2007. All Rights Reserved.

3.1.9 Proximity Modifiers

Basic proximity modifiers are defined in the CQL context set .[reference]. Proximity units 'word', 'sentence', 'paragraph', and 'element' are defined there and may also be defined in other context sets. Within the CQL set they are explicitly undefined. When defined in another context set they may be assigned specific meaning.

Thus compare "prox/unit=word" with "prox/xyz.unit=word". In the first, 'unit' is a prox modifier from the CQL set, and as such its values are undefined, so 'word' is subject to interpretation by the server. In the second, 'unit' is a prox modifier defined by the xyz context set, which may assign the unit 'word' a specific meaning.

The context set xyz may define additional units, for example, 'street':

prox/xyz.unit="street"

This approach, 'prox/xyz.unit="street"', is chosen rather than 'Prox/unit=xyz.street' for the following reason. In the first case, 'unit' is a modifier defined in the xyz context set, and 'street' is a value defined for that modifier. In the second, 'unit' is a modifier from the cql context set, with a value defined in a different set. so its value would have to be one that is defined in the cql context set. This approach is chosen to avoid pairing a modifier from one set with a value from another, which can lead to unpredictable results.

3.1.10 Sorting

Queries may include explicit information on how to sort the result set generated by the search. (See result set model [reference].)

The sort specification is included at the end, and is separated by a 'sortBy' keyword. The specification consists of an ordered list of indexes, potentially with modifiers, to use as keys on which to sort the result set. If multiple keys are given, then the second and subsequent keys should be used to determine the order of items that would otherwise sort together. Each index used as a sort key has the same semantics as when it is used to search.

Modifiers may be attached to the index in the same way as to booleans and relations in the main part of the query. These modifiers may be part of any context set, including the CQL context set and the Sort context set [reference]. This is the only time when a modifier may be attached to an index. If a modifier may be used in this way it should be stated in the description of its semantics. As many types of search also require specification of term order (for example the <, > and within relations), these modifiers are often specified as relation modifiers.

Examples:

a. "cat" sortBy dc.title

b. "dinosaur" sortBy dc.date/sort.descending dc.title/sort.ascending

Copyright © OASIS® 1993–2007. All Rights Reserved.

3.1.11 Prefix Assignment

Note: The use of Prefix Maps is expected to be uncommon.

A Prefix Map may be used to assign context set names to specific identifiers in order to be sure that the server maps them in a desired fashion. It may occur at any place in the query and applies to anything below the map in the query tree. A prefix assignment is specified by: '>' shortname '=' identifier [example a]. The shortname and '=' sign may be omitted, in which case it sets a default context set for indexes [example b].

Examples:

a. > dc = "info:units/direct-current" dc.voltage > 12
This example illustrates that while Adc@ is almost always used as the prefix for the Dublin Core context set, this is not always so, as in this case it is used for the AdeepCustard@ context set.

b. > "info:units/direct-current" voltage > 12
This query has the same meaning as example a.

3.1.12 Case Sensitivity

All parts of CQL are case insensitive apart from user supplied search terms, values for modifiers and prefix map identifiers, which may or may not be case sensitive. If any case insensitive part of CQL is specified with mixed upper and lower case, it is for aesthetic purposes only.

3.2 BNF

Following is the Backus Naur Form (BNF) definition for CQL. ("::=" represents "is defined as".)

```
sortedQuery      ::=  prefixAssignment sortedQuery
                      | scopedClause ['sortby' sortSpec]
sortSpec         ::=  sortSpec singleSpec | singleSpec
singleSpec       ::=  index [modifierList]

cqlQuery         ::=  prefixAssignment cqlQuery
                      | scopedClause
prefixAssignment ::=  '>' prefix '=' uri
                      | '>' uri
scopedClause     ::=  scopedClause booleanGroup searchClause
                      | searchClause
booleanGroup     ::=  boolean [modifierList]
boolean          ::=  'and' | 'or' | 'not' | 'prox'
searchClause     ::=  '(' cqlQuery ')'
                       | index relation searchTerm
```


Copyright © OASIS® 1993–2007. All Rights Reserved.

		| searchTerm
relation	::=	comparitor [modifierList]
comparitor	::=	comparitorSymbol | namedComparitor
comparitorSymbol	::=	'=' | '>' | '<' | '>=' | '<=' | '<>' | '=='
namedComparitor	::=	identifier
modifierList	::=	modifierList modifier | modifier
modifier	::=	'/' modifierName [comparitorSymbol modifierValue]
prefix, uri, modifierName, modifierValue, searchTerm, index	::=	term
term	::=	identifier | 'and' | 'or' | 'not' | 'prox' | 'sortby'
identifier	::=	charString1 | charString2
charString1	:=	*Any sequence of characters that does not include any of the following:* *whitespace* *((open parenthesis)* *) (close parenthesis)* *=* *<* *>* *"" (double quote)* */* *If the final sequence is a reserved word, that token is returned instead. Note that '.' (period) may be included, and a sequence of digits is also permitted. Reserved words are 'and', 'or', 'not', and 'prox' (case insensitive). When a reserved word is used in a search term, case is preserved.*
charString2	:=	*Double quotes enclosing a sequence of any characters except double quote (unless preceded by backslash (\)). Backslash escapes the character following it. The resultant value includes all backslash characters except those releasing a double quote (this allows other systems to interpret the backslash character). The surrounding double quotes are not included.*

3.3 Context Sets

CQL is so-named ("Contextual Query Language") because it is founded on the concept of searching by semantics or context, rather than by syntax. The same search may be performed in a different way on very different underlying data structures in different servers, but the important thing is that both servers

Copyright © OASIS® 1993–2007. All Rights Reserved.

understand the intent behind the query. In order for multiple communities to define their own semantics, CQL uses context sets in order to ensure cross-domain interoperability.

Context sets permit CQL users to create their own indexes, relations, relation modifiers and boolean modifiers without risk of choosing the same name as someone else and thereby having an ambiguous query. All of these four aspects of CQL must come from a context set, however there are rules for determining the prevailing default if one is not supplied. Context sets allow CQL to be used by communities in ways that the designers could not have foreseen, while still maintaining the same rules for parsing which allow interoperability.

When defining a new context set, it is necessary to provide a description of the semantics of each item within it. While context sets may contain indexes, relations, relation modifiers and boolean modifiers, there is no requirement that all should be present; in fact it is expected that most context sets will only define indexes.

Each context set has a unique identifier, a URI. When sending the context set in a query, a short form is used. These short names may be sent as a mapping within the query itself, or be published by the recipient of the query in some protocol dependent fashion. The prefix 'cql' is reserved for the base CQL context set, but authors may wish to recommend a short name for use with their set.

An index, relation, or modifier qualified by a context is represented in the form *prefix*.*value*, where *prefix* is a short name for a unique context set identifier.

Copyright © OASIS® 1993–2007. All Rights Reserved.

4 The searchRetrieve operation

The searchRetrieve operation is the main operation. It allows the client to submit a search and retrieve for matching records from the server.

4.1 Request Parameters

Name	Occurence	Description
operation	mandatory	The string: 'searchRetrieve'.
responseFormat	optional	The schema in which the response is to be supplied. If this parameter is omitted, the SR2.0 schema is assumed (as described in 4.1.2.) Other possible values are ‘atom1.0’, ‘rss2.0’, and ‘html’.
version	mandatory	The version of the request, and a statement by the client that it wants the response to be less than, or preferably equal to, that version. See .
query	mandatory	Contains a query expressed in CQL to be processed by the server. See CQL .
startRecord	optional	The position within the sequence of matched records of the first record to be returned. The first position in the sequence is 1. The value supplied MUST be greater than 0. The default value if not supplied (and if records are present in the response) is 1.
maximumRecords	optional	The number of records requested to be returned.. Default value if not supplied is determined by the server. The server MAY return less than this number of records, for example if there are fewer matching records than requested, but MUST NOT return more than this number of records.
recordPacking	optional	A string to determine how the record should be escaped in the response. Defined values are 'string' and 'xml'. The default is 'xml'. See .
recordSchema	optional	The schema in which the records MUST be returned. The value is the URI identifier for the schema or the short name for it published by the server. The default value if not supplied is determined by the server. See Record Schemas .
resultSetTTL	optional	The number of seconds for which the client requests that the result set created should be maintained. The server MAY choose not to fulfill this request, and may respond with a different number of seconds. If not supplied then the server will determine the value. See .

Copyright © OASIS® 1993–2007. All Rights Reserved.

stylesheet	optional	A URL for a stylesheet. The client requests that the server simply return this URL in the response. See .
extraRequestData	optional	Provides additional information for the server to process. See .

Example:

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

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

4.2 Response Parameters

The response to a searchRetrieve request is an XML document. The table below provides a summary and description of the elements provided by the XML document. The "Type" column indicates either an XML Schema type ("xsd:") or a type defined within the schema.

Name	Type	Occurrence	Description
version	*xsd:string*	Mandatory	The version of the response. This MUST be less than or equal to the version requested by the client. See .
numberOfRecords	*xsd:integer*	Mandatory	The number of records matched by the query. If the query fails this MUST be 0.
resultSetId	*xsd:string*	Optional	The identifier for a result set that was created through the execution of the query. See .
resultSetIdleTime	*xsd:integer*	Optional	The number of seconds after which the created result set will be deleted. The result set may also become unavailable before this. See .
records	*sequence of <record>*	Optional	A sequence of records (or surrogate diagnostics) matched by the query,. See .
nextRecordPosition	*xsd:integer*	Optional	The next position within the result set following the final returned record. If there are no remaining records, this field MUST be omitted
diagnostics	*sequence of <diagnostic>*	Optional	A sequence of non surrogate diagnostics generated during execution. See Diagnostics .
extraResponseData	*<xmlFragment>*	Optional	Additional information returned by the server. See .
echoedSearch	*<echoedSearch*	Optional	The request parameters echoed back to

Copyright © OASIS® 1993–2007. All Rights Reserved.

RetrieveRequest	*RetrieveRequest>*		the client in a simple XML form. See .

4.3 Version: the “version” Parameter

In any actively developed protocol or piece of software, there is a concern about interoperability between different versions. This protocol defines an explicit interoperability mechanism, with precisely defined semantics. The mechanism defined allows for clients and servers using different versions to interact without protocol level errors. Versions will always be recorded as strings of the format 'major.minor' where major and minor are independent integers.

All operations have a version parameter, with the exception of the parameterless form of the explain request. [See Explain operation]. For example:

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

The version parameter on a request both indicates the version of the request and is a statement by the client that it wants the response to be less than, or preferably equal to, that version. The version parameter in the response message is the version of the response. If the server cannot supply a response in that version or lower, then it must return a diagnostic. If possible this diagnostic would be in the version requested or lower, but that is not a requirement. Here are some examples of how this works in practice. If a 2.0 client asks a 1.1 server for a 2.0 response, then the server is able to respond with a 1.1 response as it is lower than version 2.0. If a 1.1 client asks a 2.0 server for a 1.1 response then the server is able to reduce its response version to accommodate the client. If a 1.1 client asks a 1.1 server for a 1.1 response, then there is no version mismatch and the server is able to accommodate the request. Version 1.0 was an experiment, and has been officially deprecated. Version 1.0 does not have a version parameter in any of the requests or responses and hence cannot be considered to be part of this version interoperability system. If a client requests version 1.0, then the server may return a 1.0 response but is under no obligation to do so.

4.4 Records

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

The records parameter in the response is a sequence of record elements, each of which contains either a record or a surrogate diagnostic explaining why that particular record could not be transferred. If the requested record schema is unknown or the record cannot be rendered in that schema, then the server MUST return a diagnostic.

4.4.1 Record Parameters

Each record element is structured into the following elements:

Name	Type	Occurence	Description
recordSchema	*xsd: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

Copyright © OASIS® 1993–2007. All Rights Reserved.

			URI.See Record Schemas
recordPacking	*xsd:string*	mandatory	The packing used in recordData, as requested by the client or the default. See below.
recordData	*<stringOrXmlFragment>*	mandatory	The record itself, either as a string or embedded XML
recordIdentifier	*xsd: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	*xsd:positiveInteger*	optional	The position of the record within the result set. See
extraRecordData	*<xmlFragment>*	optional	Any additional information to be transferred with the record. See .

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

```
<record>
    <recordSchema>info:srw/schema/1/dc-v1.1</recordSchema>
    <recordPacking>xml</recordPacking>
    <recordData>
        <srw_dc:dc xmlns:srw_dc="info:srw/schema/1/dc-v1.1">
            <dc:title>This is a Sample Record</dc:title>
        </srw_dc:dc>
    </recordData>
    <recordPosition>1</recordPosition>
    <extraRecordData>
        <rel:score xmlns:rel="info:srw/extensions/2/rel-1.0"> 0.965 </rel:score>
    </extraRecordData>
</record>
```

4.4.2 Record Packing

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

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

Copyright © OASIS® 1993–2007. All Rights Reserved.

4.5 Result Sets

Support of persistent result sets is not assumed. Thus it is not assumed that a result set created by one request may necessarily be accessed by a client in a subsequent request. The server is expected to state whether or not it supports persistent result sets, and if so the result set model described is required.

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

Even if the server does not make result sets available for public manipulation, the following model is also important to understand in order to allow a single request to both match records and then sort them.

4.5.1 Result Set Model

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

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

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

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

4.5.2 resultSetId

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

Copyright © OASIS® 1993–2007. All Rights Reserved.

4.5.3 ResultSet Idle Time

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

4.6 Diagnostics

Sometimes things go wrong. In these cases the server is obliged to report that something went wrong, by sending a diagnostic record explaining what happened. A list of diagnostics is supplied in Annex XXX and additional diagnostics may be added.

4.6.1 Diagnostic Categories: Fatal vs. Non-fatal, and Surrogate Vs. Non-Surrogate

Diagnostics fall into two categories, 'fatal' and 'non-fatal'. A fatal diagnostic is one in which the execution of the request cannot proceed and no records are available to return. For example, if the client supplied an invalid query there is nothing that the server can do. A non-fatal diagnostic on the other hand 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 also divided into two categories 'surrogate' and 'non-surrogate'. Surrogate diagnostics take the place of a record. For example if the second of three records was not available in the requested schema, then the response would include the first record, a surrogate diagnostic explaining that the second record is not available, and then the final record. Non-surrogate, non-fatal diagnostics are diagnostics saying that while some or all the records are available, something else went wrong. For example the requested sorting algorithm might not be available.

Surrogate diagnostics occur in the 'records' parameter of the response (they take the place of the record for which they are a surrogate). Non-surrogate records, both fatal and non-fatal, occur in the 'diagnostics' parameter.

To summarize: A surrogate diagnostic replaces a record; a non-surrogate diagnostic refers to the response at large and is supplied in addition to the records. A non-surrogate diagnostic may be fatal or non-fatal. So the following combinations are possible:

1. fatal (implicitly non-surrogate)
2. surrogate (implicitly non-fatal)
3. non-fatal, non-surrogate

4.6.2 Diagnostic Schema

Diagnostics are returned in a very simple schema which has only three elements, 'uri', 'details' and 'message'.

The required 'uri' field is a URI, identifying the particular diagnostic. When the URI begins with "info:srw/diagnostic/1/" *(for example, 'info:srw/diagnostic/1/7')* then the diagnostic is from the diagnostic list below. The 'details' part contains information specific to the diagnostic, format as specified by the individual diagnostic definition. The 'message' field contains a human readable message to be displayed. Only the uri field is required, the other two are optional.

Copyright © OASIS® 1993–2007. All Rights Reserved.

It is recommended for all diagnostics that the final section should be a distinguishing integer *(for example 'http://srw.cheshire3.org/diagnostics/1'*)

The identifier for the diagnostic schema is: info:srw/schema/1/diagnostics-v1.1

Name	***Type***	***Occurence***	***Description***
uri	*xsd:anyURI*	*Mandatory*	*The diagnostic's identifying URI.*
details	*xsd:string*	*Optional*	*Any supplementary information available, often in a format specified by the diagnostic*
message	*xsd: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.*

Examples

Non-surrogate, fatal diagnostic:

```
<diagnostics>
   <diagnostic xmlns="http://www.loc.gov/zing/srw/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>
```

Surrogate, non-fatal diagnostic:

```
<records>
    <record>
      <recordSchema> info:srw/schema/1/diagnostics-v1.1</recordSchema>
      <recordData>
          <diagnostic xmlns="http://www.loc.gov/zing/srw/diagnostic/">
              <uri>info:srw/diagnostic/1/65</uri>
              <message>Record deleted by another user.</message>
```


Copyright © OASIS® 1993–2007. All Rights Reserved.

```
            </diagnostic>
        </recordData> </record> ...
 </records>
```

4.7 Extensions: the “extraRequestData’, ‘extraResponseData’, and xtraRecordData’ Parameters

Messages in all of the operations**,** both in the request and in the response, have a field in which additional information may be provided. This is a built in extension mechanism where profiles may specify a schema for what to include in this section without requiring the developers to change the basic messages and thus render their implementation uninteroperable with other servers and clients. It is expected that if there is sufficient demand for a particular piece of additional information, that piece of information will be migrated into the protocol in a later version. In this way, only implemented and useful features will be added in future versions, rather than features that just seem like a good idea.

Via GET or POST, the name for an extension parameter must begin with 'x-': lower case x followed by hyphen. The protocol will never include an official parameter with a name beginning with 'x-', and hence this will never clash with a mainstream parameter name. It is recommended that the parameter name be 'x-' followed by an identifier for the namespace for the extension, again followed by a hyphen, followed by the name of the element within the namespace. For example

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

Note that this convention does not guarantee uniqueness since the parameter 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 parameter "x-info*NNN*-*XXX*" where *NNN* is the 'info:srw' authority string, and XXX is the name of the parameter. Extension names MUST never be assigned with this form except by the proper authority for the given 'info' namespace. Response Every response has an extraResponseData section. This section can include any well-formed XML, and hence servers can include namespaced XML fragments within it in order to convey information back to the client. The extension MUST supply a namespace and the element names with which to do this, if feedback to the client is necessary. For example:

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

Semantics: If the server does not understand a piece of information in an extension parameter, it may silently ignore it. This is unlike many other request parameters, where if the server does not implement that particular feature it MUST respond with a diagnostic. If the particular request requires some confirmation that it has been carried out rather than ignored, then the profile designer should include a field in the response. The semantics of parameters in the request may not be modified by extensions. For example, a *x-qt-queryType* parameter could not change query to be an SQL query, as a server that does not understand the extension would expect the query to be in CQL, and thus be unable to parse it. Instead, the extension should create a new parameter for the SQL query. The semantics of parts of the response may be modified by extensions. The response semantics may be changed in this way only if the client specifically requests the change. Clients should also expect to receive the regular semantics, as servers are at liberty to ignore extensions, and hence it is recommended that this not be done. ExtraResponseData may be sent that is not directly associated with the request. For example it may

Copyright © OASIS® 1993–2007. All Rights Reserved.

contain cost information regarding the query or information on the server or database supplying the results. This data must, however, have been requested. As the request may be echoed, the server must be able to transform the parameters into their XML form. If it encounters an unrecognized parameter, the server may either make its best guess as to how to transform the parameter, or simply not return it at all. It should not, however, add an undefined namespace to the element as this would invalidate the response. If the content of the parameter is an XML structure, then the extension designer should also specify how to encode this structure in a URL. This may simply be to escape all of the special characters, but the designer could also create a string encoding form with rules as to how to generate the XML in much the same fashion as the relationship between CQL and XCQL.

echoedSearch

4.8 Echoing the Request: The "echoedSearchRetrieveRequest" Parameter

Very thin clients, such as a web browser with a stylesheet as above, may not have the facility to record the query that generated the response it has just received. In order to prevent clients having to maintain this information, the server may echo the request back to the client along with the response. There are no request elements associated with this functionality. There is one response element per operation in which the request is echoed. The name of this is the name of the response element, prefixed by echoed. The parameters are rendered into XML.

4.8.1 xQuery

xQuery is an additional parameter for searchRetrieve and scan, which has the query rendered in XCQL [reference]. This has two benefits:

a. The client can use XSLT or other XML manipulation to modify the query without having a CQL query parser.

b. The server can return extra information specific to the clauses within the query. See the next section on extensions for more information.

4.8.2 baseUrl

A server can include is own base URL in the echoed request. This allows the client to easily 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.

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>
```


Copyright © OASIS® 1993–2007. All Rights Reserved.

```
    <baseUrl>http://z3950.loc.gov:7090/voyager</baseUrl>
</echoedSearchRetrieveRequest>
```

4.9 Stylesheets: the ‘stylesheet’ Parameter

In order to render the response, "thin" clients may provide a stylesheet to turn the response XML into a natively renderable format, often HTML or XHTML. This allows a web browser, or other application capable of rendering stylesheets, to act as a dedicated client without requiring any further application logic. The parameter on the response enables a client to use this stylesheet to also have the request it just made available without any client side logic. OperationsAll operations, other than the parameterless explain request, have the stylesheet parameter. The value of the parameter is the URL of the stylesheet to be included in the response. This URL is to be included 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. If the server cannot fulfill this request it must supply a diagnostic . This parameter may not be used via SOAP. It is a SOAP error to return a stylesheet, and hence an error to request one. If this parameter is not supplied, then the server can, at its discretion, include a default stylesheet. The default stylesheet URL may be included in the explain document. For example, upon receiving the request ...

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

...the server must include the following as beginning of the response:

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


Copyright © OASIS® 1993–2007. All Rights Reserved.

5 Scan Operation

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

The index to be browsed and the start point within it is given in the scanClause parameter as a complete index, relation, term clause in CQL. The relation and relation modifiers may be used to determine the format of the terms returned. For example 'dc.title any fish' will return a list of keywords, whereas 'dc.title exact fish' would return a list of full title fields. Range relations, such as <, >=, within and so forth, are prohibited for use with scan, and diagnostic 'info:srw/diagnostic/1/19' should be returned. See below for a clarifying example.

The term given in the clause is the position within the ordered list of terms at which to start, however see the responsePosition parameter below for more information. If the empty term is given, then even if searching for it is unsupported by the server, it may be interpreted as the beginning of the term list.

5.1 Request Parameters

Name	Occurence	Description
operation	mandatory	The string: 'scan'.
version	mandatory	The version of the request, and a statement by the client that it wants the response to be less than, or preferably equal to, that version. See .
scanClause	mandatory	The index to be browsed and the start point within it, expressed as a complete index, relation, term clause in CQL. See CQL .
responsePosition	optional	The position within the list of terms returned where the client would like the start term to occur. If the position given is 0, then the term should be immediately before the first term in the response. If the position given is 1, then the term should be first in the list, and so forth up to the number of terms requested plus 1, meaning that the term should be immediately after the last term in the response, even if the number of terms returned is less than the number requested. The range of values is 0 to the number of terms requested plus 1. The default value is 1.
maximumTerms	optional	The number of terms which the client requests be returned. The actual number returned may be less than this, for example if the end of the term list is reached, but may not be more. The explain record for the database may indicate the maximum number of terms which the server will return at once. All positive integers are valid for this parameter. If not specified, the default is server determined.
stylesheet	optional	A URL for a stylesheet. The client requests that the server

Copyright © OASIS® 1993–2007. All Rights Reserved.

		simply return this URL in the response. See .
extraRequestData	optional	Provides additional information for the server to process. See .

559 Example:

560 http://myserver.com/sru?operation=scan&version=1.2&scanClause=dc.title = frog
561 &responsePosition=1&maximumTerms=25

562 5.2 Response Parameters

563

Name	Type	Occurence	Description
version	xsd:string	mandatory	The version of the response. This MUST be less than or equal to the version requested by the client. See .
terms	sequence of <term>	optional	A sequence of terms which match the request. See
diagnostics	sequence of <diagnostic>	Optional	A sequence of non surrogate diagnostics generated during execution. See Diagnostics .
extraResponseData	xmlFragment	Optional	Additional information returned by the server. See .
echoedScanRequest	<echoedScanRequest>	Optional	The request parameters echoed back to the client in a simple XML form. See .

564

565 5.3 Terms

Name	Type	Occurence	Description
value	xsd:string	mandatory	The term, exactly as it appears in the index.
numberOfRecords	xsd:nonNegativeInteger	optional	The number of records which would be matched if the index in the request's scanClause was searched with the term in the 'value' field.
displayTerm	xsd:string	optional	A string to display to the end user in place of the term itself. For example this might add back in diacritics or capitalisation which do not appear in the index.
whereInList	xsd:string	optional	A flag to indicate the position of the term within the complete term list. It must be one of the following values:

Copyright © OASIS® 1993–2007. All Rights Reserved.

			'first' (the first term), 'last' (the last term), 'only' (the only term) or 'inner' (any other term)
extraTermData	xmlFragment	optional	Additional information concerning the term. See .

5.4 Example Scan Response

```
<sru:scanResponse xmlns:srw="http://www.loc.gov/zing/srw/"
                  xmlns:diag="http://www.loc.gov/zing/srw/diagnostic/"
                  xmlns:myServer="http://myServer.com/">
  <sru:version>1.1</sru:version>
  <sru:terms>

    <sru:term>
      <sru:value>cartesian</sru:value>
      <sru:numberOfRecords>35645</sru:numberOfRecords>
      <sru:displayTerm>Carthesian</sru:displayTerm>
    </sru:term>

    <sru:term>
      <sru:value>carthesian</sru:value>
      <sru:numberOfRecords>2154</sru:numberOfRecords>
      <sru:displayTerm>CarthÉsian</sru:displayTerm>
    </sru:term>

    <sru:term>
      <sru:value>cat</sru:value>
      <sru:numberOfRecords>8739972</sru:numberOfRecords>
      <sru:displayTerm>Cat</sru:displayTerm>
    </sru:term>

    <sru:term>
      <sru:value>catholic</sru:value>
      <sru:numberOfRecords>35</sru:numberOfRecords>
      <sru:displayTerm>Catholic</sru:displayTerm>
      <sru:whereInList>last</sru:whereInList>
      <sru:extraTermData>
        <myserver:ID>4456888</myserver:ID>
      </sru:extraTermData>
    </sru:term>
```


Copyright © OASIS® 1993–2007. All Rights Reserved.

```
601
602       </sru:terms>
603
604       <sru:echoedScanRequest>
605            <sru:version>1.1</sru:version>
606            <sru:scanClause>dc.title="cat"</sru:scanClause>
607           <sru:responsePosition>3</sru:responsePosition>
608           <sru:maximumTerms>3</sru:maximumTerms>
609           <sru:stylesheet>http://myserver.com/myStyle</sru:stylesheet>
610       </sru:echoedScanRequest>
611   </sru:scanResponse>
```

612
613

Copyright © OASIS® 1993–2007. All Rights Reserved.

6 The Explain Facility

The Explain Facility allows a client to retrieve a description of the resources and services available at a server. It can then be used by the client to self-configure and provide an appropriate interface to the user. The record is in XML and follows the ZeeRex Schema. There are two methods for getting the explain record:

a. **Via the Explain Operation**
   See 6.1.

b. **Via the http GET request at the base URL for the service**
   This can be considered a searchRetrieve request, no parameters, and hence a default recordPacking of 'xml', with no extraRequestData and leaving it up to the server to determine the version of the response. Otherwise, the response is identical to an explainResponse message.

6.1 Explain Operation

6.1.1 Request Parameters

Name	occurence	Description
operation	Mandatory	The string: 'explain'.
version	Mandatory	The version of the request, and a statement by the client that it wants the response to be less than, or preferably equal to, that version. See .
recordPacking	Optional	A string to determine how the explain record should be escaped in the response. Defined values are 'string' and 'xml'. The default is 'xml'. See .
stylesheet	Optional	A URL for a stylesheet. The client requests that the server simply return this URL in the response. See .
extraRequestData	Optional	Provides additional information for the server to process. See .

4.3.2 Response Parameters

Name	Type	occurence	Description
version	xsd:string	Mandatory	The version of the response. This MUST be less than or equal to the version requested by the client. See …
record	record	Mandatory	A single Explain record, wrapped in the record metadata fields. See .
extraResponseData	xmlFragment	Optional	Additional information returned by the server. >> See .
echoedExplainRequest	<echoedExplainRequest>	Optional	The request parameters echoed back to the client in a simple XML form. >> See

Copyright © OASIS® 1993–2007. All Rights Reserved.

7 XML and WSDL Files

XML and WSDL files for the above defined operations will be provided in the published version of this standard.

This current discussion document is based on SRU. The XML and WSDL files for SRU version 1.1 can be found at:

http://www.loc.gov:8081/standards/sru/sru1-1archive/xml-files.html

Copyright © OASIS® 1993–2007. All Rights Reserved.

8 Transports

8.1 HTTP Get Binding

The client may send a request via the HTTP GET method. A URL is constructed and sent to the server with fixed parameter names with fixed meanings. When unicode characters need to be encoded, there are some additional constraints, discussed below.

The response must be XML conforming to the response schema of the operation. HTTP GET can thus be described as the simplest case of XML over HTTP.

An example of what might pass over the wire:

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

8.1.1 Syntax

A request (when transported via HTTP GET) is a URI as described in RFC 3986 (See). Specifically it is an HTTP URL (as described in section 3.3 of RFC 1738) ; however there are some further notes about character encoding below, and uses the standard & separated *key=value* encoding for parameters in the query part of the URI.

The parameters for the query section of the URL (the information following the question mark) of the various operations are described in their own sections.

8.1.2 Encoding Issues

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. This is normally relevant only to the query parameter of the searchRetrieve operation and the scanClause parameter of the scan operation

1. .Convert the value to UTF-8.
2. Percent-encode characters as necessary within the value. See
3. Construct the 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. If in doubt, percent-encode.

Example

Consider the following parameter:

Copyright © OASIS® 1993–2007. All Rights Reserved.

667 query=dc.title =/word *kirkegård*

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

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

671 The following characters must be percent encoded:

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

673 - the '/', percent encoded as %2F

674 - the spaces, percent encoded as %20

675 - the 'å'. Its UTF-8 representation is C3A5, two octets, and correspondingly it is represented in a
676 URI as two characters percent encoded as %C3%A5.

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

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

679 8.1.3 Server Procedure

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

685 *Note:*

686 *RFC 1738 is obsoleted by RFC 3986. However, RFC 1738 describes the 'http:' URI scheme; RFC 3986*
687 *does not, instead indicating that a separate document will be written to do so, but it has not yet been*
688 *written. So currently there is no valid, normative reference for the 'http:' URI scheme, and so the obsolete*
689 *RFC 1738 is referenced. When there is a valid, normative reference, it will be listed here.*
690

691 8.2 HTTP Post Binding

692 Instead of constructing a URL, the parameters may be sent via POST to the server. The Content-type
693 header MUST be set to 'application/x-www-form-urlencoded'. Compare to 'text/xml' - via SOAP below,
694 which can be used to distinguish the two transports at the same end point.

695 POST has several benefits over GET for transferring the request to the server. Primarily the issues with
696 character encoding in URLs are removed, and an explicit character set can be submitted in the Content-
697 type HTTP header. Secondly, very long queries might generate a URL for HTTP GET that is not
698 acceptable by some web servers or client. This length restriction can be avoided by using POST.
699

700 The response via POST is identical to that of GET, an xml document.

Copyright © OASIS® 1993–2007. All Rights Reserved.

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
```

8.3 SOAP Binding

This is a binding to the SOAP recommendation of the W3C . In this transport, the request is encoded in XML and wrapped in some additional SOAP specific elements. The response is the same XML as via GET or POST, but wrapped in additional SOAP specific elements.

The incremental 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.

8.3.1 SOAP Requirements

- Clients and servers MUST support SOAP version 1.1, and MAY support version 1.2 or higher. This requirement is allow as much flexibility in implementation as possible.
- 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 the POST binding thus have three content-type headers to consider.

The specification tries to adhere to the Web Services Interoperability recommentations.

8.3.2 SOAP 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.

Example SOAP request:

```
<SOAP:Envelope xmlns:SOAP="http://schemas.xmlsoap.org/soap/envelope/">
    <SOAP:Body>
```


Copyright © OASIS® 1993–2007. All Rights Reserved.

```
734         <SRW:searchRetrieveRequest xmlns:SRW="http://www.loc.gov/zing/srw/">
735            <SRW:version>1.1</SRW:version>
736            <SRW:query>dinosaur</SRW:query>
737            <SRW:startRecord>1</SRW:startRecord>
738            <SRW:maximumRecords>1</SRW:maximumRecords>
739            <SRW:recordSchema>info:srw/schema/1/mods-v3.0</SRW:recordsSchema>
740         </SRW:searchRetrieveRequest>
741      </SOAP:Body>
742   </SOAP:Envelope>
```

743

744 8.3.3 Extension Parameters via SOAP

745 Via SOAP, the extension parameters are XML structures. The request parameters are identified by their
746 full namespace, and the name of the parameter is the name of the XML element. Even if there is only one
747 piece of additional information supplied, it must be within a namespaced XML element. This is in order to
748 ensure that servers can distinguish a parameter from one extension from another. For example:

```
749 <extraRequestData>
750  <theo:onSearchFail xmlns:theo="info:srw/extension/4/searchextensions">
751  scan
752  </theo:onSearchFail>
753  </extraRequestData>
```

754

Copyright © OASIS® 1993–2007. All Rights Reserved.

A. The CQL Context Set

Normative Annex

The CQL context set defines a set of indexes, relations and relation modifiers. The indexes supplied are 'utility' indexes which are generally ~~u~~ useful across all applications of the language. These utility indexes are for instances when CQL is required to express a concept not directly related to the records, or for indexes applicable in practically every context. The reserved name for this context set is: cql

The identifier for this context set is: info:srw/cql-context-set/1/cql-v1.2

A.1 Indexes

- **resultSetId**
  A search clause may be a result set id. This is a special case, where the index and relation are expressed as "cql.resultSetId =" and the term is the result set id returned by the server in the 'resultSetId' parameter of the searchRetrieve response. It may be used by itself in a query to refer to an existing result set from which records are desired. It may also be used in conjunction with other resultSetId clauses or other indexes, combined by boolean operators. The semantics of resultSetId with relations other than "=" is undefined. The semantics of resultSetId with scan is also undefined.

  Example:
  cql.resultSetId = "5940824f-a2ae-41d0-99af-9a20bc4047b1"
  Match the result set with the given identifier.

- **allRecords**
  A special index which matches every record available. Every record is matched no matter what values are provided for the relation and term, but the recommended syntax is: cql.allRecords = 1. The semantics for scanning allRecords is not defined.

  Example:
  cql.allRecords = 1 NOT dc.title = fish
  Search for all records that do not match 'fish' as a word in title.

- **allIndexes**
  Alias: anywhere
  The 'allIndexes' index will result in a search equivalent to searching all of the indexes (in all of the context sets) that the server has access to. The semantics for scanning allIndexes is not defined.

  Example:
  cql.allIndexes = fish
  If the server had three indexes title, creator and date, then this would be the same as title = fish or creator = fish or date = fish

- **anyIndexes**
  Alias: serverChoice
  The 'anyIndexes' index allows the server to determine how to search for the given term. The

Copyright © OASIS® 1993–2007. All Rights Reserved.

server may choose one or more indexes in which to search, which may or may not be generally available via CQL. It may choose a different index to search every time, based on the term for example, and hence may not produce consistent results via scan.

This is the default when the index and relation is omitted from a search clause. The relation used when the index is omitted is '='.
Examples:
cql.anyIndexes = fish
Search in any one or more indexes for the term fish

- **keywords**
  The keywords index is an index of terms from the record, determined by the server as being generally descriptive or meaningful to search on. It might include the full text of a document, descriptive metadata fields, or anything else generally useful to search as an initial entry point to the data. Exactly which fields make up this index is determined by the server, however the choice must be consistent, unlike anyIndexes above, when the choice can be different for different searches.

  Example:
  cql.keywords any/relevant "code computer calculator programming"
  Search in descriptive locations for the given term

A.2 Relations

A.2.1 Implicit Relations

These relations are defined as such in the grammar of CQL. The cql context set only defines their meaning, rather than their existence.

- =
  This is the default relation, and the server can choose any appropriate relation or means of comparing the query term with the terms from the data being searched. If the term is numeric, the most commonly chosen relation is '==='. For a string term, either 'adj' or '==' as appropriate for the index and term.

  Examples:

  - animal.numberOfLegs = 4
    The recommended server choice for this example is '=='
  - dc.identifer = "gb 141 staff a-m"
    The recommended server choice for this example is '=='
  - *dc.title = "lord of the rings"*
    The recommended server choice for this example is 'adj'
  - *dc.date = "2004 2006"*
    The recommended server choice for this example is 'within'

- ==
  This relation is used for exact equality matching. The term in the data is exactly equal to the term in the search.
  Examples:

Copyright © OASIS® 1993–2007. All Rights Reserved.

- dc.identifier == "gb 141 staff a-m"
  Search for the string 'gb 141 staff a-m' in the identifier index.
- dc.date == "2006-09-01 12:00:00"
  Search for the given datestamp.
- animal.numberOfLegs == 4
  Search for animals with exactly 4 legs.

- <>

  This relation means 'not equal to' and matches anything which is not exactly equal to the search term.
  Examples:
  - dc.date <> 2004-01-01
    Search for any date except the first of January, 2004
  - dc.identifier <> ""
    Search for any identifier which is not the empty string.

- <, >, <=,>=

  These relations retain their regular meanings as pertaining to ordered terms (less than, greater than, less than or equal to, greater than or equal to).
  Examples:
  - dc.date > 2006-09-01
    Search for dates after the 1st of September, 2006
  - animal.numberOfLegs < 4
    Search for animals with less than 4 legs.

A.2.2 Defined Relations

These relations are defined as being widely useful as part of a default context set.

- adj

  This relation is used for phrase searches. All of the words in the search term must appear, and must be adjacent to each other in the record in the order of the search term. The query could also be expressed using the PROX boolean operator.
  Examples:
  - dc.title adj "day in the life"
    Search for the phrase 'lord of the rings' somewhere in the title.
  - dc.description adj "blue shirt"
    Search for 'blue' immediately followed by 'shirt' in the description.

- *all, any*

  These relations may be used when the term contains multiple items to indicate "all of these items" or "any of these items". These queries could be expressed using boolean AND and OR respectively. These relations have an implicit relation modifier of 'cql.word', which may be changed by use of alternative relation modifiers.
  Examples:
  - dc.title all "day life"
    Search for both day and life in the title.

Copyright © OASIS® 1993–2007. All Rights Reserved.

- dc.description any "computer calculator"
  Search for either computer or calculator in the description.

- *within*

  Within may be used with a search term that has multiple dimensions. It matches if the database's term falls completely within the range, area or volume described by the search term, inclusive of the extents given.
  Examples:
  - dc.date within "2002 2003"
    Search for dates between 2002 and 2003 inclusive.
  - animal.numberOfLegs within "2 5"
    Search for animals that have 2,3,4 or 5 legs.

- encloses

  Conversely, encloses is used when the index's data has multiple dimensions. It matches if the database's term fully encloses the search term.
  Examples:
  - xyz.dateRange encloses 2002
    Search for ranges of dates that include the year 2002.
  - *geo.area encloses "45.3, 19.0"*
    Search for any area that encloses the point 45.3, 19.0
    This example needs more work

A.3 Relation Modifiers

A.3.1 Functional Modifiers

- stem

  The server should apply a stemming algorithm to the words within the term. For example such that computing and computer both match the stem of 'compute'.

- relevant

  The server should use a relevancy algorithm for determining matches and the order of the result set.

- phonetic

  The server should use a phonetic algorithm for determining words which sound like the term.

- fuzzy

  The server should be liberal in what it counts as a match. The exact details of this are left up to the server, but might include permutations of character order, off-by-one for numerical terms and so forth.

- partial

  When used with within or encloses, there may be some section which extends outside of the term. This permits for the database term to be partially enclosed, or fall partially within the search

Copyright © OASIS® 1993–2007. All Rights Reserved.

term.

- ignoreCase, respectCase
  The server is instructed to either ignore or respect the case of the search term, rather than its default behavior (which is unspecified). This modifier may be used in sort keys to ensure that terms with the same letters in different cases are sorted together or separately, respectively.

- ignoreAccents, respectAccents
  The server is instructed to either ignore or respect diacritics in terms, rather than its default behavior (which is unspecified, but respectAccents is the recommended default). This modifier may be used in sort keys, to ensure that characters with diacritics are sorted together or separately from those without them.

- locale=value
  The term should be treated as being from the specified locale. Locales will in general include specifications for whether sort order is case-sensitive or insensitive, how it treats accents, and so forth. The default locale is determined by the server. The value is usually of the form C, french, fr_CH, fr_CH.iso88591 or similar. This modifier may be used in sort keys.

Examples:

- dc.title any/stem "computing disestablishmentarianism"
  Find the local stemmed form of 'computing' and 'disestablishmentarianism', and search for those stems in the stemmed forms of the terms in titles.

- person.phoneNumber =/fuzzy "0151 795-4252"
  Search for a phone number which is something similar to '0151 795-4252' but not necessarily exactly that number.

- "fish" sortBy dc.title/ignoreCase
  Search for 'fish', and then sort the results by title, case insenstively.

- dc.title within/locale=fr "l m"
  Find all titles between l and m, ensure that the locale is 'fr' for determining the order for what is between l and m.

A.3.2 Term-format Modifiers

These modifiers specify the format of the search term to ensure that the server performs the correct comparison. These modifiers may all be used in sort keys.

- word
  The term should be broken into words, according to the server's definition of a 'word'

- string
  The term is a single item, and should not be broken up.

Copyright © OASIS® 1993–2007. All Rights Reserved.

- isoDate
  Each item within the term conforms to the ISO 8601 specification for expressing dates.

- number
  Each item within the term is a number.

- uri
  Each item within the term is a URI.

- oid
  Each item within the term is an ISO object identifier, dot-separated format.

Examples:

- dc.title =/string Jaws
  Search in title for the string 'Jaws', rather than Jaws as a word. (Equivalent to the use of == as the relation)

- zeerex.set ==/oid "1.2.840.10003.3.1"
  Search for the given OID as an attribute set.

- squirrel sortby numberOfLegs/number
  Search for squirrel, and sort by the numberOfLegs index ensuring that it is treated as a number, not a string. (eg '2' would sort after '10' as a string, but before it as a number)

A.3.3 Masking

- *masked*
  This is a default modifier, that is, it is assumed if omitted. To explicitly request this functionality, add 'cql.masked' as a relation modifier. The following masking rules and special characters apply for search terms, unless overridden in a profile via a relation modifier.
  - *****
    A single asterisk (*) is used to mask zero or more characters.
  - **?**
    A single question mark (?) is used to mask a single character, thus N consecutive question-marks means mask N characters.
  - **^**
    Carat/hat (^) is used as an anchor character for terms that are word lists, that is, where the relation is 'all' or 'any', or 'adj'. It may not be used to anchor a string, that is, when the relation is '==' (string matches are, by default, anchored). It may occur at the beginning or end of a word (with no intervening space) to mean right or left anchored."^" has no special meaning when it occurs within a word (not at the beginning or end) or string but must be escaped nevertheless.
  - ****
    Backslash (\) is used to escape '*', '?', quote (") and '^' , as well as itself. Backslash not followed immediately by one of these characters is an error.

Copyright © OASIS® 1993–2007. All Rights Reserved.

Examples:

- dc.title = c*t
  Matches words that start with c and end in t

- dc.title adj "*fish food*"
  Matches a word that ends in fish, followed by a word that starts with food

- dc.title = c?t
  Matches a three letter word that starts with c and ends in t.

- dc.title adj "^cat in the hat"
  Matches 'cat in the hat' where it is at the beginning of the field

- dc.title any "^cat ^dog rat^"
  Matches cat at the beginning, dog at the beginning or rat at the end

- dc.title == "\"Of Couse\", she said"
  Escape internal double quotes within the term.

- ***unmasked***
  Do not apply masking rules, all characters are literal.

- ***substring***
  The 'substring' modifier may be used to specify a range of characters (first and last character) indicating the desired substring within the field to be searched. The modifier takes a value, of the form "start:end" where start and end obey the following rules:
  - Positive integers count forwards through the string, starting at 1. The first character is 1, the tenth character is 10.
  - Negative integers count backwards through the string, with -1 being the last character.
  - Both start and end are inclusive of that character.
  - If omitted, start defaults to 1 and end defaults to -1.

Examples:

- dc.title =/substring="-5:" title
- marc.008 =/substring="1:6" 920102
- dc.title =/substring=":" "The entire title"
- dc.title =/substring="2:2" h

- regexp
  The term should be treated as a regular expression. Any features beyond those found in

Copyright © OASIS® 1993–2007. All Rights Reserved.

modern POSIX regular expressions are considered to be server dependent. This modifier overrides the default 'masked' modifier, above. It may be used in either a string or word context.

Example:

dc.title adj/regexp "(lord|king|ruler) of th[ea] r.*s"

Match lord or king or ruler, followed by of, followed by the or tha, followed by r plus zero or more characters plus s

A.4 Booleans

A context set cannot define booleans, as these are defined by the CQL grammar. A context set can define semantics of the booleans defined by the CQL grammar, and this context set defines the following semantics.

- **AND**
  The combination of two sets of records with AND will result in the set of records that appear in both of the sets.

- **OR**
  The combination of two sets of records with OR will result in the set of records that appear in either or both of the sets. It is therefore inclusive OR, not exclusive OR.

- **NOT**
  The combination of two sets of records with NOT will result in the set of records that appear in the left set, but not in the right hand set. It cannot be used as a unary operator.

- **PROX**
  The prox (short for proximity) boolean operator allows for the relative locations of the terms to be used in order to determine the resulting set of records. The semantics of when a match occurs is defined by the modifiers or defaults for those modifiers, as described below.

A.5 Boolean Modifiers

The CQL context set defines four boolean modifiers, which are only used with the prox boolean operator.

- **distance** <symbol> <value>
  The distance that the two terms should be separated by.
  - Symbol is one of: **<, >**, **<=**, **>=**, **=**, **<>**
    If the modifier is not supplied, it defaults to <=.
  - Value is a non-negative integer. If the modifier is not supplied, it defaults to 1 when unit=word, or 0 for all other units.

- unit= <value>
  The type of unit for the distance.
  - Value is one of: '**paragraph** ,**sentence**, **word**, **element.** The default is 'word'. These values are explicitly undefined. They are subject to interpretation by the server.

Copyright © OASIS® 1993–2007. All Rights Reserved.

See “Note About Proximity Units” below.

- unordered
  The order of the two terms is unimportant. This is the default.

- **ordered**
  The order of the two terms must be as per the query.

Examples:

- cat prox/unit=word/distance>2/ordered hat
  Find 'cat' where it appears more than two words before 'hat'

- cat prox/unit=paragraph hat
  Find cat and hat appearing in the same paragraph (distance defaulting to 0) in either order (unordered default)

- zeerex.set = cql prox/unit=element/distance=0 zeerex.index = resultSetId
  Find the cql context set in the same element as the index name resultSetId. E.g. search for cql.resultSetIds

Note about Proximity Units

As noted above proximity units 'paragraph', 'sentence', 'word' and 'element' are explicitly undefined when used by the CQL context set. Other context sets may assign them specific values.

Thus compare "prox/unit=word" with "prox/xyz.unit=word". In the first, 'unit' is a prox modifier from the CQL set, and as such its values are undefined, so 'word' is subject to interpretation by the server. In the second, 'unit' is a prox modifier defined by the xyz context set, which may assign the unit 'word' a specific meaning.

Other context sets may define additional units, for example, 'street':

prox/xyz.unit="street"

Note that this approach, 'prox/xyz.unit="street"', is preferable to 'Prox/unit=xyz.street'. In the first case, 'unit' is a modifier defined in the xyz context set, and 'street' is a value defined for that modifier. In the second, 'unit' is a modifier from the cql context set, with a value defined in a different set. so its value would have to be one that is defined in the cql context set. Pairing a modifier from one set with a value from another is not a good practice.

Copyright © OASIS® 1993–2007. All Rights Reserved.

B. Diagnostics

Normative Annex

The diagnostics below are defined for use with the following 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: i*nfo:srw/diagnostic/1/2*). The details format is 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. Some of the diagnostics from earlier versions of the standards have been deprecated, however they are still listed here, suitably marked, for reference. For additional explanation of these diagnostics, see .xxx

General Diagnostics			
Number	**Description (additional description in notes below)**		**Details Format**
1	General system error		Debugging information (traceback)
2	System temporarily unavailable		
3	Authentication error		
4	Unsupported operation		
5	Unsupported version		Highest version supported
6	Unsupported parameter value		Name of parameter
7	Mandatory parameter not supplied		Name of missing parameter
8	Unsupported Parameter		Name of the unsupported parameter

Copyright © OASIS® 1993–2007. All Rights Reserved.

Diagnostics Relating to CQL			
Number	***Description (additional description in notes below)***		***Details Format***
10	Query syntax error		
12	Too many characters in query		Maximum supported
13	Invalid or unsupported use of parentheses		Character offset to error
14	Invalid or unsupported use of quotes		Character offset to error
15	Unsupported context set		URI or short name of context set
16	Unsupported index		Name of index
18	Unsupported combination of indexes		Space delimited index names
19	Unsupported relation		Relation
20	Unsupported relation modifier		Value
21	Unsupported combination of relation modifers		Slash separated relation modifiers
22	Unsupported combination of relation and index		Space separated index and relation
23	Too many characters in term		Length of longest term
24	Unsupported combination of relation and term		Space separated relation and term
26	Non special character escaped in term		Character incorrectly escaped
27	Empty term unsupported		
28	Masking character not supported		
29	Masked words too short		Minimum word length
30	Too many masking characters in term		Maximum number supported
31	Anchoring character not supported		
32	Anchoring character in		Character offset

Copyright © OASIS® 1993–2007. All Rights Reserved.

	unsupported position		
33	Combination of proximity/adjacency and masking characters not supported		
34	Combination of proximity/adjacency and anchoring characters not supported		
35	Term contains only stopwords		Value
36	Term in invalid format for index or relation		
37	Unsupported boolean operator		Value
38	Too many boolean operators in query		Maximum number supported
39	Proximity not supported		
40	Unsupported proximity relation		Value
41	Unsupported proximity distance		Value
42	Unsupported proximity unit		Value
43	Unsupported proximity ordering		Value
44	Unsupported combination of proximity modifiers		Slash separated values
46	Unsupported boolean modifier		Value
47	Cannot process query; reason unknown		
48	Query feature unsupported		Feature
49	Masking character in unsupported position		the rejected term
50	Result sets not supported		
51	Result set does not exist		Result set identifier
52	Result set temporarily unavailable		Result set identifier
53	Result sets only supported for retrieval		
55	Combination of result sets with		

Copyright © OASIS® 1993–2007. All Rights Reserved.

	search terms not supported		
58	Result set created with unpredictable partial results available		
59	Result set created with valid partial results available		
60	Result set not created: too many matching records		Maximum number
Diagnostics Relating to Records			
Number	***Description (additional description in notes below)***		***Details Format***
61	First record position out of range		
64	Record temporarily unavailable		
65	Record does not exist		
66	Unknown schema for retrieval		Schema URI or short name
67	Record not available in this schema		Schema URI or short name
68	Not authorised to send record		
69	Not authorised to send record in this schema		
70	Record too large to send		Maximum record size
71	Unsupported record packing		
72	XPath retrieval unsupported		
73	XPath expression contains unsupported feature		Feature
74	Unable to evaluate XPath expression		

Copyright © OASIS® 1993–2007. All Rights Reserved.

Diagnostics Relating to Sorting			
Number	***Description (additional description in notes below)***		***Details Format***
80	Sort not supported		
82	Unsupported sort sequence		Sequence
83	Too many records to sort		Maximum number supported
84	Too many sort keys to sort		Maximum number supported
86	Cannot sort: incompatible record formats		
87	Unsupported schema for sort		URI or short name of schema given
88	Unsupported path for sort		XPath
89	Path unsupported for schema		XPath
90	Unsupported direction		Value
91	Unsupported case		Value
92	Unsupported missing value action		Value
93	Sort ended due to missing value		

1153

Copyright © OASIS® 1993–2007. All Rights Reserved.

Diagnostics relating to Stylesheets			
Number	***Description (additional description in notes below)***		***Details Format***
110	Stylesheets not supported		
111	Unsupported stylesheet		URL of stylesheet
Diagnostics relating to Scan			
Number	**Description (additional description in notes below)**		**Details Format**
120	Response position out of range		
121	Too many terms requested		maximum number of terms

1154

1155 **Notes**

1156

1157

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 searchRetrieve where an undefined operation is sent.
5	general	Unsupported version	Currently only version 1.1 is defined and so thisëgnostic 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

Copyright © OASIS® 1993–2007. All Rights Reserved.

			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'.
10	query	Query syntax error	The query was invalid, but no information is given for exactly what was wrong with it. Eg. dc.title foo fish (The reason is that foo isn't a valid relation in the default context set, but the server isn't telling you this for some reason)
12	query	Too many characters in query	The length (number of characters) of the query exceeds the maximum length supported by the server.
13	query	Invalid or unsupported use of parentheses	The query couldn't be processed due to the use of parentheses. Typically either that they are mismatched, or in the wrong place. Eg. (((fish) or (sword and (b or) c)
14	query	Invalid or unsupported use of quotes	The query couldn't be processed due to the use of quotes. Typically that they are mismatched Eg. "fish'
15	query	Unsupported context set	A context set given in the query isn't known to the server. Eg. foo.title any fish
16	query	Unsupported index	The index isn't known, possibly within a context set. Eg. dc.author any sanderson (dc has a creator index, not author)
18	query	Unsupported combination of indexes	The particular use of indexes in a boolean query can't be processed. Eg. The server may not be able to do title queries merged with description queries.

Copyright © OASIS® 1993–2007. All Rights Reserved.

19	query	Unsupported relation	A relation in the query is unknown or unsupported. Eg. The server can't handle 'within' searches for dates, but can handle equality searches.
20	query	Unsupported relation modifier	A relation modifier in the query is unknown or unsupported by the server. Eg. 'dc.title any/fuzzy starfish' when fuzzy isn't supported.
21	query	Unsupported combination of relation modifers	Two (or more) relation modifiers can't be used together. Eg. dc.title any/cql.word/cql.string "star fish"
22	query	Unsupported combination of relation and index	While the index and relation are supported, they can't be used together. Eg. dc.author within "1 5"
23	query	Too many characters in term	The term is too long. Eg. The server may simply refuse to process a term longer than a given length.
24	query	Unsupported combination of relation and term	The relation cannot be used to process the term. Eg dc.title within "sanderson"
26	query	Non special character escaped in term	Characters may be escaped incorrectly Eg "\a\r\n\s"
27	query	Empty term unsupported	Some servers do not support the use of an empty term for search or for scan. Eg: dc.title > ""
28	query	Masking character not supported	A masking character given in the query is not supported. Eg. The server may not support * or ? or both
29	query	Masked words too short	The masked words are too short, so the server won't process them as they would likely match too many terms. Eg. dc.title any *
30	query	Too many masking characters in term	The query has too many masking characters, so the server won't process them. Eg. dc.title any "???a*f??b* *a?"
31	query	Anchoring character not supported	The server doesn't support the anchoring character (^) Eg dc.title = "^jaws"
32	query	Anchoring character in unsupported position	The anchoring character appears in an invalid part of the term, typically the middle of a word. Eg dc.title any "fi^sh"
33	query	Combination of proximity/adjacenc	The server cannot handle both adjacency (= relation for words) or proximity (the

Copyright © OASIS® 1993–2007. All Rights Reserved.

		y and masking characters not supported	boolean) in combination with masking characters. Eg. dc.title = "this is a titl* fo? a b*k"
34	query	Combination of proximity/adjacency and anchoring characters not supported	Similarly, the server cannot handle anchoring characters.
35	query	Term contains only stopwords	If the server does not index words such as 'the' or 'a', and the term consists only of these words, then while there may be records that match, the server cannot find any. Eg. dc.title any "the"
36	query	Term in invalid format for index or relation	This might happen when the index is of dates or numbers, but the term given is a word. Eg dc.date > "fish"
37	query	Unsupported boolean operator	For cases when the server does not support all of the boolean operators defined by CQL. The most commonly unsupported is Proximity, but could be used for NOT, OR or AND.
38	query	Too many boolean operators in query	There were too many search clauses given for the server to process.
39	query	Proximity not supported	Proximity is not supported at all.
40	query	Unsupported proximity relation	The relation given for the proximity is unsupported. Eg the server can only process = and > was given.
41	query	Unsupported proximity distance	The distance was too big or too small for the server to handle, or didn't make sense. Eg 0 characters or less than 100000 words
42	query	Unsupported proximity unit	The unit of proximity is unsupported, possibly because it is not defined.
43	query	Unsupported proximity ordering	The server cannot process the requested order or lack thereof for the proximity boolean
44	query	Unsupported combination of proximity modifiers	While all of the modifiers are supported individually, this particular combination is not.
46	query	Unsupported boolean modifier	A boolean modifier on the request isn't supported.
47	query	Cannot process query; reason unknown	The server can't tell (or isn't telling) you why it can't execute the query, maybe it's a bad query or maybe it requests an unsupported capability.

Copyright © OASIS® 1993–2007. All Rights Reserved.

48	query	Query feature unsupported	the server is able (contrast with 47) to tell you that something you asked for is not supported.
49	query	Masking character in unsupported position	eg, a server that can handle xyz* but not *xyz or x*yz
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

Copyright © OASIS® 1993–2007. All Rights Reserved.

67	records	Record not available in this schema	The record schema is known, but this particular record cannot be transformed into it.
68	records	Not authorised to send record	This particular record requires additional authorisation in order to receive it.
69	records	Not authorised to send record in this schema	The record can be retrieved in other schemas, but the one requested requires futher authorisation.
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.
80	sort	Sort not supported	the server cannot perform any sort; that is the server only returns data in the default sequence.
82	sort	Unsupported sort sequence	The particular sequence of sort keys is not supported, but the keys may be supported individually.
83	sort	Too many records to sort	used when the server will only sort result sets under a certain size and the request returned a set larger than that limit.
84	sort	Too many sort keys to sort	the server can accept a sort statement within a request but cannot deliver as requested, e.g. the server can sort by a maximum of 2 keys only such as "title" and "date" but was requested to sort by "title", "author" and "date".
86	sort	Cannot sort: incompatible record formats	The result set includes records in different schemas and there is insufficient commonality among the schemas to enable a sort.
87	sort	Unsupported schema for sort	the server does not support sort for records in a particular schema, e.g. it supports sort for records in the DC

Copyright © OASIS® 1993–2007. All Rights Reserved.

			schema but not in the ONIX schema.
88	sort	Unsupported path for sort	the server can accept a sort statement within a request but cannot deliver as requested, e.g. the server can deliver in title or date sequence but subject was requested.
89	sort	Path unsupported for schema	The path given cannot be generated for the schema requested. For example asking for /record/fulltext within the simple Dublin Core schema
90	sort	Unsupported direction	the server can accept a sort statement within a request but cannot deliver as requested, e.g. the server can deliver in ascending only but descending was requested.
91	sort	Unsupported case	the server can accept a sort statement within a request but cannot deliver as requested, e.g. the server's index is single case so sorting case sensitive is unsupported
92	sort	Unsupported missing value action	the server can accept a sort statement within a request but cannot deliver as requested. For example, the request includes a constant that the server should use where a record being sorted lacks the data field but the server cannot use the constant to override its normal behavior, e.g. sorting as a high value.
93	sort	Sort ended due to missing value	missingValue of >abort=
110	stylesheet	Stylesheets not supported	The 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.
120	scan	Response position out of range	The request includes a position in response that is not valid for the list. For example a request indicates a response position = 15 and maximum terms = 20, meaning that it wants a response to include 15 entries before the term, plus the term, then another 4. The server would return this diagnostic if there were not 15 previous entries.
121	scan	Too many terms	Say you ask for 500 terms and the server has a (fixed) maximum of 300. It would

Copyright © OASIS® 1993–2007. All Rights Reserved.

		requested	supply a value of '300' for details. If 'details' is not supplied, this might mean that the server doesn't have a fixed maximum and was just unable to deliver all the requested terms.

Copyright © OASIS® 1993–2007. All Rights Reserved.

C. NISO Z39.92 (ZeeRex)

Normative Annex

ZeeRex Summary

- The protocol attribute on the serverInfo element MUST have the value: SRU
- The transport attribute on the serverInfo element MUST be one of: http or https
- The method attribute on the serverInfo element MUST be a space separated list, comprising any number of the following values: GET POST SOAP
- The database element within serverInfo MUST contain the path section of the URL to the server, without the first / and up to the ?
- The set element within indexInfo is used to define the short names of context sets.
- Indexes are described by including the name of the index in the name element within map, and the short name for the context set in the set attribute on that element.
- The schemaInfo section is used to described the schemas supported by the server.

Examples

The following URLs would all retrieve the explain document:

- http://myserver.com/cgi/mysru?operation=explain&version=1.1&recordPacking=xml
- http://myserver.com/cgi/mysru?
- http://myserver.com/cgi/mysru

The corresponding response from the server would be:

```
<sru:explainResponse xmlns:sru="http://www.loc.gov/zing/srw/">
<sru:version>1.1</sru:version>
        <sru:record>
                <sru:recordPacking>XML</sru:recordPacking>
                <sru:recordSchema>http://explain.z3950.org/dtd/2.1/</sru:recordSchema>
                <sru:recordData>

                        <zr:explain xmlns:zr="http://explain.z3950.org/dtd/2.1/">
                        <zr:serverInfo protocol="SRU" version="1.2" transport="http"
                                                method="GET POST SOAP">
```


Copyright © OASIS® 1993–2007. All Rights Reserved.

```
                        <zr:host>myserver.com</zr:host>
                        <zr:port>80</zr:port>
                        <zr:database>cgi/mysru</zr:database>
                    </zr:serverInfo>
                    <zr:databaseInfo>
                        <title lang="en" primary="true">SRU Test Database</title>
                    </zr:databaseInfo>
                    <zr:indexInfo>
                        <zr:set name="dc" identifier="info:srw/cql-context-set/1/dc-v1.1"/>
                        <zr:index>
                            <zr:map><zr:name set="dc">title</zr:name></zr:map>
                        </zr:index>
                    </zr:indexInfo>
                    <zr:schemaInfo>
                        <zr:schema name="dc" identifier="info:srw/schema/1/dc-v1.1">
                            <zr:title>Simple Dublin Core</zr:title>
                        </zr:schema>
                    </zr:schemaInfo>
                    <zr:configInfo>
                        <zr:default type="numberOfRecords">1</zr:default>
                        <zr:setting type="maximumRecords">50</zr:setting>
                        <zr:supports type="proximity"/>
                    </zr:configInfo>
                </zr:explain>

            </sru:recordData>
            </sru:record>
    </sru:explainResponse>
```


Copyright © OASIS® 1993–2007. All Rights Reserved.

D. OpenSearch

This Annex describes the [OpenSearch] binding for the Search interface. The intent is to encourage servers to support OpenSearch.

The existing (legacy) OpenSearch specification can be found at
http://www.opensearch.org/Specifications/OpenSearch/1.1/Draft_3

(Note:this URL will be updated if the specification is updated prior to publication of this standard.)

This annex is intended to be compatible with that (legacy) specification. However the protocol as specified by this standard supports OpenSearch functionality (thougn not in a manner that is interoperable with the legacy OpenSearch spec) and OpenSearch users are encouraged to migrate to this standard.

D.1 OpenSearch Description Document

In order for an OpenSearch client to initiate a search on a server that implements the Search Web Services interface, the server must the server must expose its supported search queries by declaring them in its OpenSearch Description document.

A server should serve an OpenSearch description document as a resource at the following URL relative to the base URL for the server:

```
#Open Search description document URL
/search/opensearchdescription.xml
```

Listing 1: OpenSearch Description Document Relative URL

The following listing shows an example of an OpenSearch Description document for a server that supports a Search WS interface.

There is a name associated with the Search WS specification (alias “sws”).

```
<?xml version=" 1.0" encoding="UTF-8"?>
<OpenSearchDescription
  xmlns="http://a9.com/-/spec/opensearch/1.1/"
  xmlns:sws="urn:oasis:names:tc:search-ws:param-query:xsd:1.0"
  >

  <ShortName>Example Search Engine for Search WS</ShortName>
  <Description>Use any OpenSearch client to search this engine using template
URLs declared in Url elements below.</Description>
  <Tags>Search WS OASIS</Tags>
  <Contact>admin@example.com</Contact>

  <!--Template URL for FindById query-->
  <Url type="application/sws+xml"
```


Copyright © OASIS® 1993–2007. All Rights Reserved.

```
template="http://example.com/search?query={sws:query}&responseFormat={sws:resp
onseFormat?}&version={sws:version}&startRecord={sws:startRecord?}&maximumRecor
ds={sws:maximumRecords?}&language={sws:language?}"/>

</OpenSearchDescription>
```

Listing 2: Example: OpenSearch Description Document

D.2 OpenSearch URL Template

Within the OpenSearch Description document the most important elements are the URL templates. Each URL template declares a parameterizes query supported by the server and defines a template for the URL to invoke it.

When describing URL template we will use a URL structure as follows:

http_URL = "http:" "//" host [":" port] [abs_path_prefix “/” abs_path_suffix ["?" queryOption]]

The rules for defining a URL template structure are as follows:

- The URL template MAY be implementation specific upto and including the abs_path_prefix
- The URL template MUST have a abs_path_suffix of “/search”
- Each primitive Request parameter is mapped to a queryOption that must be declared
  - If the parameter is equivalent to a standard OpenSearch parameter then it should use a parameter name as an unqualified name as defined in Table 1:

Request Parameter	OpenSearch Parameter
maximumRecords	count
startRecord	startIndex
language	language

Table 1: Mapping of Response Parameters to Standard OpenSearch Parameters

  - If the parameter has no equivalent standard OpenSearch parameter then its should declare its template as a qualified name (Qname) within the “sws” namespace defined by this specification
  - Each template parameter should indicate if it is optional according to the specification for the Request parameter within this specification

D.3 OpenSearch Response Elements

A server must return the search response in a format specified by the responseFormat parameter of the scanOperation Request. If no responseFormat parameter is specified then it MUST return the response in the default responseFormat defined by this specification If the server does not support the requested

Copyright © OASIS® 1993–2007. All Rights Reserved.

1290 responseFormat then it MUST send an error response with a diagnostic of UnsupportedFormatException
1291 appropriate error status code as defined in Appendix B.

1292
1293
1294

Copyright © OASIS® 1993–2007. All Rights Reserved.

E. Authentication, Authorization, and Access Control

NON-NORMATIVE ANNEX

Authentication, authorization, and access control are outside the scope of this standard. This non-normative Annex provides suggested approaches.

Some business models may impose requirements, for example, to ensure that one user does not modify another's result sets, to allow a server to restrict a user to a pre-determined number of searches before charges are imposed, or to limit the number of concurrent searches for a user or number within a certain time frame. Or, on the other hand, if it can be demonstrated that a search has led directly to a sale, then the user may receive a commission. Another example is to enable the service to track how different users use the system, possibly to enforce acceptable usage policies.

This section aims to discuss the various methods in which different users may be authenticated in an interoperable manner. In a stateless environment, or one where the ability to track individual users is not important, this can be ignored without peril.

There are several technical methods by which distinct users may be identified, from IP address to additional header information to SSL. The different methods create additional requirements and function at various levels of success.

E.1 Authentication

A server SHOULD support HTTP Basic authentication, HTTP/S Digest authentication for all bindings that use the HTTP transport protocol.

If a server supports single sign-on using an external authentication authority it SHOULD do so using SAML 2.0 protocols, profiles and bindings.

E.2 Authorization and Access Control

A server is free to use any suitable mechanisms for authorization and access control of a client connection. A server MUST remove any results from the search result set that the client is not authorized to retrieve. A server SHOULD masking any parts of a result if the user is not authorized to see that part of the result. An example is the use of '*' to mask a password value.

E.3 IP Address

Users may be differentiated by the IP address from which they are connecting to the server. Unfortunately this is unreliable at best due to the increasing use of web proxy systems -- there may be many users all of which appear to be coming from the same IP address due to a proxy. The advantage is that it is completely transparent to the client and hence the user, so for a small service may be appropriate.

E.4 Basic Authentication

Basic Authentication is the fairly simple method used in many web servers to authenticate users against a list or database -- the client is required to send a username and password. This is a very easy-to-configure method to authenticate users, however it does not allow for users that are not authenticated --

Copyright © OASIS® 1993–2007. All Rights Reserved.

every request must have a valid user and password sent or it will be rejected. This model is appropriate for a paid-for service or one which is used only by a set of known individuals, but is less appropriate for a service which may be used by anyone.

E.5 Secure Sockets

SSL is an encrypted version of HTTP (https) and hence is more secure than basic authentication alone as the traffic cannot be easily intercepted. For financial transations this is certainly appropriate as the user is already known in advance and every care for the data must be taken. However for every day services that may be used by anyone, it is a very complex solution.

E.6 Additional Message Data

The preferred method for identifying users while still allowing non-authenticated access is by the inclusion of an additional field in the extraRequestData and extraResponseData fields. This method allows the server to chose when authentication is required (for example only if a resultset is needed) and when it can continue to act in a stateless fashion. This may be appropriate for any sort of transaction with the exception of cases when the data should be conveyed in an encrypted fashion, in which case SSL should be used as well.

The recommended name for this field is *authenticationToken*, and hence *x-authenticationToken* when it is passed on the URL-. If the server sends back one of these tokens with a response, then the client should return it in the same fashion in any subsequent request to allow the server to know that the requests should be considered to be from the same user.

Further business logic may be required to manipulate these tokens. For example a separate SOAP service may be required to distribute the tokens on request, to delete tokens when they've finished being used or to enable the sharing of such tokens between users to allow shared access to result sets..

The URI for the namespace for this extension is **info:srw/extension/2/auth-1.0**

E.7 Web Services Security and Security Assertion Markup Language (SAML) Security Tokens

The OASIS committee has defined the Web Services Security (WS-Security) Standard[1][1] which specifies how different security tokens, signature formats and encryption technologies are to be used for secure Web service, in terms of end-to-end message content security, and not just transport-level security. The signatures and security tokens are defined within the <wsse:Security> element of a SOAP message header.

---

[1][1] http://www.oasis-open.org/committees/download.php/16790/wss-v1.1-spec-os-SOAPMessageSecurity.pdf

Copyright © OASIS® 1993–2007. All Rights Reserved.

An important security token format used by WS-Security is the SAML Security Token. The SAML standard[2][2] specifies how authentication and attribute assertions about a subject can be made from a trusted source. In a federated environment, these assertions would typically come from a trusted authentication and attribute authority (referred to as the Identity Provider), and allow the receiver (often referred to as the Service Provider) to make authorization decisions based on these attributes. The assertions are signed to ensure integrity, and can optionally be encrypted to preserve confidentiality.

By leveraging WS-Security and SAML tokens, an SRU/SRW search service (acting as a Service Provider in the SAML scenario above) can authenticate and authorize a search request simply based on the SAML assertions contained in its request header. This allows the search service to be available to a much wider set of users from many different security domains, not just the traditional local security domain.

[2][2] http://www.oasis-open.org/committees/tc_home.php?wg_abbrev=security#samlv20

Copyright © OASIS® 1993–2007. All Rights Reserved.