Search Web Services – The OASIS SWS Technical Committee Work: The Abstract Protocol Definition, OpenSearch Binding, and SRU/CQL 2.0

[作者] Ray Denenberg
[单位] Library of Congress

[摘要] The OASIS Search Web Services Technical Committee is developing search and retrieval web services, integrating various approaches under a unifying model, an Abstract Protocol Definition. SRU/CQL and OpenSearch are the two approaches featured by the current work, and we hope that additional protocols will be similarly integrated into this model.The model provides for the development of bindings. Three bindings will be developed by the Committee: SRU 1.2, OpenSearch, and SRU 2.0. These three are so-called "static" bindings; they are human-readable documents. The first two are simply renderings of the respective existing specifications. The SRU 2.0 binding however is a major new version of SRU, and there will also be a new version of the companion query language, CQL 2.0. The model also defines the concept of a "dynamic" binding, a machine-readable description file that a server provides for retrieval by a client that may then dynamically configure itself to access that server. The premise of the dynamic binding concept is that any server – even one that pre-dated the concept – need only provide a self-description in order to be accessible. A client will be able to access the server simply by reading and interpreting the description and, based on that description, formulating a request (including a query) and interpreting the response. Of course, the premise behind this concept is a standard description language, and that will also be part of the OASIS work.

[关键词] Search Web Services, Work

1. Introduction

In 2007 OASIS [1] created the Search Web Services Technical Committee. Its charge is to define search and retrieval web services based on various current technologies, most notably, Search and Retrieval via URL (SRU) [2] – along with its companion Contextual Query Language (CQL) [3] – and OpenSearch [4].

SRU (together with CQL) and OpenSearch are at different ends of the search/retrieval complexity spectrum, and these two protocols are the focus of the OASIS work. (A major part of the work will be revisions to SRU and CQL.) The foundation of the work will be an Abstract Protocol Definition (APD), a reference model by which these and other protocols can be described.

1.1 Overview of this Article

Section 1.2 provides some brief historical notes about SRU and CQL. Section 2 describes the Abstract Protocol Definition and how concrete specifications, called bindings, are derived.

A binding may be static or dynamic. The Committee's focus is on static bindings, but the APD lays the foundation for dynamic bindings. The nature of the description language, used for dynamic bindings, is briefly described and an example is provided in section 2.2.

The OpenSearch binding is described in section 3 and the SRU 2.0 binding in section 4; new features of SRU and CQL are described in detail. Finally, the projected schedule of work is outlined in section 5. This is followed by a concluding section.

1.2 Historical Notes on SRU and CQL

SRU was originally conceived as one of two companion protocols, SRW [5] and SRU. In SRW (Search and Retrieve Web Service) messages are conveyed using XML over HTTP via SOAP. With SRU, clients send messages to servers by URL. SRW is no longer presented as a separate protocol but rather as a variation of SRU, referred to as "SRU via HTTP SOAP" [6].

Collectively, this suite (the two protocols and CQL) was originally called "Z39.50 Next Generation", and subsequently, "Z39.50 International Next Generation" ("ZING"). These names are no longer used.

Development of SRW/SRU/CQL began in 2000. An experimental version, 1.0, was released in November 2002, and the first official version, 1.1, in February 2004. A minor revision, 1.2, was released in 2007, in preparation for the work that was soon to begin within OASIS, to include development of SRU/CQL version 2.0.

2 The Abstract Protocol Definition and its Bindings

Although version 2.0 of SRU and CQL will be the most visible specifications produced by the Committee, the foundation of the work will be the Abstract Protocol Definition, specifying abstract search request parameters and abstract search response elements. These abstractions provide the framework for the definitions of Application Protocol Bindings.

2.1 Bindings

A binding is a realization of the APD: it is a concrete protocol specification derived by mapping abstractions within the APD to real objects and concepts. The first such binding developed by the Committee will be for SRU Version 1.2. The SRU 1.2

specification already exists, and the binding is being developed in part as proof-of-concept. It describes how the abstractions defined in the APD are realized in SRU 1.2, and it supplies the abstract-to-concrete mappings of request parameters and response elements.

2.1.1 Example of an Abstraction and its Realization

As an example, the APD describes the following abstraction:

A server exposes a datastore for access by a remote client for purposes of search and retrieval. The datastore is a collection of units of data. Such a unit is referred to as an item.

Thus the APD introduces the abstractions datastore and item. The SRU binding explains that a 'datastore' as described in the APD is realized as a database in the binding, and that an 'item' is realized as a database record.

Corresponding to the item abstraction, the APD defines the abstract parameter 'maximumItems' (the number of items requested to be returned). The analogous parameter in the SRU 1.2 binding is 'maximumRecords'.

As another example, the APD defines the abstract parameter 'responseItemType', and the corresponding SRU 1.2 parameter is 'recordSchema'.

2.1.2 Static and Dynamic Bindings

A binding may be static or dynamic. The SRU 1.2 binding is a static binding: it is specified by a human-readable document. The concept of a static binding isn't very interesting for SRU 1.2, because an SRU 1.2 specification already exists. An SRU 2.0 binding will be more interesting, since no SRU 2.0 specification yet exists. (The OASIS SRU 2.0 binding will be the first.)

In contrast to a static binding, which takes the form of a human-readable document, a dynamic binding takes the form of a machine-readable description file that a server provides for retrieval by a client, which may then dynamically configure itself to access that server. It is not a specification or standard; its only manifestation is a file on a server.

The premise of the dynamic binding concept is that any server - even one that pre-dated the concept - need only provide a self-description. It need make no other change in order to be accessible. A client will be able to access a server that provides a description simply by reading and interpreting the description and, based on that description, formulating a request (including a query) and interpreting the response.

Of course, the premise behind this concept is a standard description language, and that will be part of the OASIS work. In section 2.2, there is an example description file.

2.2 A Description File Example

Before looking at the example below, a word of caution: the description language has not yet been drafted, so not only is this example hypothetical, but the syntax is as well.

The hypothetical description file below contains:

- a general description (element <databaseInfo>);
- a request-formulation element (<requestInfo>); and
- a response-interpretation element (<responseInfo>).

```
<swsDescription>
<!-- -->
   <databaseInfo>
      <name>Science Fiction Database</name>
       <shortName>SciFi</shortName>
       <contact>
              <name>Ralph LeVan</name>
              <email>levan@oclc.org</email>
      </contact>
   </databaseInfo>
<!-- -->
   <requestInfo>
     <template>

http://orlabs.oclc.org/SRW/search/scifi?query={query}&version=1.2&operation=sea
rchRetrieve
&maximumRecords={maximumItems}&startRecord={startPosition}
     </template>
     <example>
           http://orlabs.oclc.org/SRW/search/scifi?query= "it's a good
life"&version=1.2
          &operation=searchRetrieve&maximumRecords=12&startRecord=1
      </example>
   </requestInfo>
<!-- -->
  <responseInfo type='xml' xmlns:srw='http://www.loc.gov/zing/srw/'>
     <numberOfItems>
         <tagpath>/srw:searchRetrieveResponse/numberOfRecords</tagpath>
```

```
    </numberOfItems>
   <item>
       <tagpath>
        /srw:searchRetrieveResponse/srw:records/srw:record/srw:recordData
       </tagpath>
   </item>
    <diagnostics>
      <tagpath>/srw:searchRetrieveResponse/srw:diagnostics</tagpath>
    </diagnostics>
  </responseInfo>
</swsDescription>
```

2.2.1 Request Formulation

Within the request-formulation element, <requestInfo>, the <template> element is of most interest. It describes how a search request is to be formulated. Worth noting in the template are:

- query={query}
- maximumRecords={maximumItems}
- startRecord={startPosition}

The latter two indicate that this server uses parameter names 'maximumRecords' and 'startRecord' respectively for the abstract parameters 'maximumItems' and 'startPosition' defined in the APD. The first simply says that the query parameter name is 'query'.

This tells the client that if you want to formulate a request where (for example):

- the query is to be "to serve man";
- the value of parameter maximumItems (as described in the APD) is to be 10; and
- the value of parameter startPosition (as described in the APD) is to be 1;

the request URL should be:

http://orlabs.oclc.org/SRW/search/scifi?query="to serve man"&version=1.2&operation=searchRetrieve&maximumRecords=10&startRecord=1

Note: 'version' and 'operation' are not abstract parameters. The inclusion of the two parameter assignments, version=1.2 and operation=searchRetrieve, means that they are to be included verbatim in the request; they have meaning to the server. They are not intended to convey any semantics as part of the template.

2.2.2 Response Information

The <responseInfo> element describes how information may be extracted from the response. For example, consider the following fragment:

```
   <numberOfItems>
<tagpath>/srw:searchRetrieveResponse/numberOfRecords</tagpath>
   </numberOfItems>
```

The value of the <tagpath> element is an XPath expression. It says that the abstract element <numberOfItems> (as described in the APD) corresponds to the element <numberOfRecords> in the response.

3 The OpenSearch Binding

The template approach used to describe request formulation was inspired by OpenSearch. The OASIS work will include an OpenSearch binding.

3.1 The OpenSearch Template

Consider the following OpenSearch URL template (an example from the Opensearch specification) as expressed by the <url> element in an OpenSearch description:

```
  <url type="application/rss+xml"
  template="http://example.com/?q={searchTerms}&pw={startPage?}&format=rss"/>
```

First, consider the parameter assignments:

- q={searchTerms}
- pw={startPage?}

'searchTerms' and 'startPage' are abstract parameters (see 3.2) defined in the OpenSearch specification.

'q={searchTerms}' is the server's way of saying "use 'q' for the parameter 'searchTerms'". Similarly, 'pw={startPage?}' says "use 'pw' for the parameter startPage'". (The question mark, '?', means that the parameter is optional.)

To put it another way, the server is saying via the template:

"A request where the value of searchTerms is (for example) 'cat' and the value of startPage is (for example) '1' looks like this:

http://example.com/?q=cat&pw=1&format=rss

The attribute assignment in the <url> element:

type="application/rss+xml"

associates this mime type (application/rss+xml) with this template. That means that this particular template is used if you want the response to be in that format. There may be additional <url> elements with different templates for other response formats. In this case the template includes the string:

```
format=rss
```

which is what this particular server expects to see in a request if RSS is the expected response format. This is not an abstract parameter assignment; the server is just including the exact parameter assignment string verbatim in the template, in order to say "if you want the response to be of mime type 'application/rss+xml', then include the string 'format=rss'".

3.2 OpenSearch Viewed as an Abstract Protocol Definition

OpenSearch defines abstract request parameters. In this sense it is similar to the OASIS Abstract Protocol Definition. (It does not, however, define abstract response elements.)

The OASIS OpenSearch binding maps the APD abstract parameters to OpenSearch parameters. And since these OpenSearch parameters are themselves abstract parameters (see example, next paragraph), the OpenSearch binding is not a static binding. But neither is it a dynamic binding. We call it an intermediate binding.

For example, the APD abstract parameter 'query' corresponds to the OpenSearch parameter 'searchTerms'. However, 'searchTerms' is itself an abstract parameter - an OpenSearch abstract parameter.

This is different from the SRU binding (described above). For SRU, where (for example) the APD abstract parameter 'startPosition' is assigned to the actual parameter 'startRecord', this means that 'startRecord' is literally the string that goes over the wire to represent the parameter. But for OpenSearch, where the APD parameter 'query' corresponds to the parameter 'searchTerms', the latter is not the name that goes over the wire, rather it is considered by OpenSearch as an abstract parameter, for which a server assigns a real parameter, 'q' for example (as in the example above).

3.3 OpenSearch Response

Common response formats such as RSS and ATOM are used in OpenSearch, providing a degree of interoperability. But these formats, in their native forms, are not fully functional response formats for a search protocol; they require specialization. Consider the following example response (adapted from the OpenSearch specification):

```
<feed xmlns="http://www.w3.org/2005/Atom"
       xmlns:opensearch="http://a9.com/-/spec/opensearch/1.1/">
   <title>Example.com Search: New York history</title>
   <link href="http://example.com/New+York+history"/>
   <updated>2003-12-13T18:30:02Z</updated>
   <author>
     <name>Example.com, Inc.</name>
   </author>
   <id>urn:uuid:60a76c80-d399-11d9-b93C-0003939e0af6</id>
   <opensearch:totalResults>4230000</opensearch:totalResults>
   <opensearch:startIndex>21</opensearch:startIndex>
   <opensearch:itemsPerPage>10</opensearch:itemsPerPage>
   <link rel="alternate" href="http://example.com/New+York+History?pw=3"
       type="text/html"/>
   <entry>
     <title>New York History</title>
     <link  href="http://www.columbia.edu/cu/
            lweb/eguids/amerihist/nyc.html"/>
     <id>urn:uuid:1225c695-cfb8-4ebb-aaaa-80da344efa6a</id>
     <updated>2003-12-13T18:30:02Z</updated>
     <content type="text">
       ... Harlem.NYC - A virtual tour and information on
       businesses ...  with historic photos of Columbia's own New York
       neighborhood ... Internet Resources for the City's History. ...
     </content>
   </entry>
 </feed>
```

This XML file is an enhanced ATOM feed - it includes non-ATOM elements, from the OpenSearch namespace: for example, 'totalResults', 'startIndex', and 'itemsPerPage'. These elements are not part of the ATOM namespace, but they are necessary in a search response. The APD defines abstract elements for each of these and maps each to its corresponding OpenSearch name in the OpenSearch binding.

Note that in contrast to the OpenSearch request, where a server may use whatever name it chooses for an abstract parameter, for the response the server is constrained to use the element name defined by the OpenSearch namespace. For the request parameters the server exposes the name via the template in the description file (as described above), but OpenSearch does not provide a similar mechanism for a server to expose local response element names.

4 The SRU 2.0 Binding

The current focus of the OASIS SWS Technical Committee is a major revision to SRU and CQL: SRU/CQL 2.0. This will come in the form of an APD binding for SRU 2.0 and a companion CQL 2.0 specification.

The Committee has sought search and retrieval requirements from various communities for various applications. These include geospatial, Learning Object Metadata (LOM), Legislative, and the SRU Implementor Group. This section discusses some of the features planned for 2.0.

4.1 Multiple Query Types

In an SRU 1.2 request, the query is always a CQL query, and it is conveyed via the parameter whose name is 'query'. Thus the CQL query "dc.title=cat" would be conveyed by the parameter assignment:

query=dc.title=cat

There is no explicit indication that it is a CQL query. It is known to be a CQL query because in SRU 1.2 all queries are CQL queries.

In SRU 2.0 other query types besides CQL may be used.

A new parameter, 'queryType', will be added. Its value might be "cql", in which case the value of the 'query' parameter will be a CQL query. Or the value of 'queryType' may be, for example, 'XQuery' [7], in which case the value of the 'query' parameter would be an XQuery string. The parameter will be optional and, if omitted, will default to CQL. (The server may override this default, via Explain.)

4.2 Proximity: Elements and Windows

In a proximity query, matching occurs when two specified terms (usually two, but sometimes more than two) bear a specified relationship to one-another. The relationship is usually based on distance. For example:

1. Find "cat" and "hat" in the same paragraph.
2. Find "cat" and "hat" in the same sentence.
3. Find "cat" and "hat" separated by exactly two words.
4. Find "cat" and "hat" separated by no more than three words.
5. Find "cat" and "hat" separated by at least three words.
6. Find "cat" followed by "hat" separated by no more than three words.
7. Find the name "adam smith" and date "1965" in the same author element.
8. Find "cat", "hat", and "rat" within a 10-word window.

Note: these examples are offered in plain English, not CQL syntax.

Except for examples 7 and 8, these are all well understood and supported in Z39.50 and CQL. For each of examples 1 through 6 there is a distance unit and value. In example 1 the unit is 'paragraph' and distance zero. In example 3 the unit is 'word' and distance is two. Each has a relationship: 'less than', 'no more than', 'exactly'. And example 6 specifies an order: 'followed by'.

Examples 7 and 8 are not well supported, and they are being addressed within the OASIS work. These are examples of two distinct problems: same element and window.

4.2.1 Same element

Although the query:

find "cat" and "hat" in the same sentence

is easily constructed in CQL, the query:

find the name "adam smith" and date "1965" in the same author element

is not. Even though it may seem on the surface that these queries are structurally similar, a "sentence" is a fairly well understood textual unit, but an "element" is a structural unit not as well-understood. (More below.)

This is a classic problem in information retrieval: finding a specific author, for example "Adam Smith born in 1965", as distinguished from "Adam Smith born in 1942".

A Boolean (AND) query such as:

author element contains "adam smith" AND author element contains "1965"

is not precise enough. There may be a record with multiple author elements, one of which is "Adam Smith b.1942" and another, "Matilda Plews b.1965", and this will result in a false match. So a simple Boolean AND query isn't sufficient to express "same element". Some sort of proximity expression is necessary, but "same element" is not a typical proximity relationship.

Nevertheless the LOM community needs to express this sort of relationship. Therefore, this is one of the problems to be solved in version 2.0.

The problem has engendered debate over abstract vs. structured querying. A fundamental assumption of CQL is that the structure of the data being searched is not known. A distinguishing feature of CQL is its abstractions, most notably, abstract indexes. If the data to be searched is known to conform to a specific XML search schema, and if the intent is to exploit that specific structure, then a structured query language, XQuery for example, should be used instead.

CQL assumes that records may for example be MODS [8] which is highly structured; they may be Dublin Core, which is relatively unstructured; or they may be some other structure. A query like "find the name 'adam smith' and date '1965' in the same author element", if it is to be a CQL query, must be constructed so that it could apply to Dublin Core data, MODS data, or any other data for which this query makes sense.

The solution must not employ structured querying. By that we mean if the structure of the data were known to be, for example, MODS, and queryable as such, a structured query language could formulate a query to search for element <name> where:

- The value of the role attribute is "author".
- The value of subelement <namePart> -- such that the value of the type attribute is"given" -- is "john".
- The value of subelement <namePart> -- such that the value of the type attribute is "family" -- is "smith"
- The value of subelement <namePart> -- where the value of the type attribute is "date" -- is "1965".

CQL, on the other hand, would address the query independent of structure, and the query might look something like:

```
bib.name ="adam smith" PROX/element=bib.author dc.date =1965
```

In this hypothetical CQL expression (see note), bib.name, bib.author, and dc.date are abstract indexes; "abstract" meaning that the server does not necessarily have indexes with these names but that these abstract indexes have known semantics sufficient for a server to map them to local real indexes.

Note: in the example CQL expression above, the syntax is hypothetical because "element" is not yet defined as a proximity qualifier.

Now this CQL expression does seem at the surface to convey structure; that is, it seems to imply that within the data there is some author field (though not necessarily with name "author") that has subfields "name" and "date". But in reality, the data being searched may be Dublin Core, and there may be a record with element:

```
<creator>Adam Smith 1965</creator>
```

And the server's matching algorithm very well might be sophisticated enough that this record is a match.

4.2.2 Window

Consider the "window" query:

```
Find "cat", "hat", and "rat" within a 10 word window
```

The query is for occurrences within an arbitrary interval of a specified size (a window of 10 words). The window query is different from the "same element" query; a window is an arbitrary interval while an element is a specific structural unit.

The window problem is further distinguished because proximity (in Z39.50 and in SRU/CQL) is an operation performed on term pairs; e.g. "Find 'cat' and 'hat' within ten words of one another". For the window query, three (or more) terms are sought. There is no way to combine pair-based proximity operations to express this query. To illustrate that fact, consider the following passage:

"The black hat fell off the black cat who ate the black rat and then put on the black hat."

"cat" and "hat" occur within 10 words of one-another, as do "cat" and "rat", and "rat" and "hat". But the three words do not occur in any 10 word window.

Consider the following, more realistic example. In a legislative database, Bills from the 110th Congress, the query:

Medical Insurance Payments

reported that 650 bills included all three words, while only one bill contained the exact phrase "medical insurance payments". In three bills, the three terms occurred near one-another (within a 20 word window).

For this example, where a searcher is interested in "medical insurance payments", a query for bills containing all three words results in an overwhelming number of extraneous matches, and a search for the exact phrase risks omitting two (of only three) potentially important results.

Furthermore, searching pair-wise on the three words: "medical insurance", "insurance payments", and "medical payments", resulted in 59, 67, and 75 hits respectively (bills where the two words occurred near one-another). So it is doubtful that these queries could have been combined in any meaningful way if "medical insurance payments" is of interest.

The window" operation will be a new feature of CQL 2.0, where several words, and a window size, may be expressed.

4.3 Alternative Response Format

In SRU 1.2, the format of the response is fixed. There is a schema defined in the specification and all responses must be supplied according to that schema. In SRU 2.0 a response may be supplied in an alternative schema, for example ATOM or RSS.

An SRU server must be able to support the SRU response schema, but the client may request an alternative schema.

4.4 Faceted Search Results

If a server supports faceted search results, SRU/CQL 2.0 will support the capability to convey the faceted results in a response. A schema will be defined for faceted results as part of SRU/CQL 2.0, and the server may supply faceted results according to that schema.

Faceted searching is illustrated in the following example.

An SRU client submits the query:

nuthatch

The server reports a result count of 50 and lists three queryable facets:

- subject
- author
- date

The user clicks on "subject" and obtains a list of subjects, and for each, the number of records that list the given subject:

- birds – 15 records
- nuthatches – 12 records
- Sitta carolinensis – 4 records

The user clicks on Sitta carolinensis which results in a new query: the original query qualified by subject="Sitta carolinensis". The resultant query is:

Nuthatch AND subject="Sitta carolinensis"

This query is sent to the server and the response contains the four records.

Similarly the user could obtain a list of authors, or a list of dates, and qualify the search by a particular author or date.

4.5 Result Size Precision

In SRU the server reports the result set size in every response. The size reported is assumed to be accurate: if the server reports that the query produced 123 results, the client infers that there are exactly 123 results. There is no provision for the server to report "approximately 123 results", "at least 123 results", "at most 123 results", or "number of results unknown".

During the past several years implementors have suggested that servers should be able to indicate or estimate the accuracy of the reported result-set size. So an element (with name something like <resultSizePrecision>) will be added to the response in SRU 2.0. The value will be a term from a controlled vocabulary, including values such as 'exact', 'unknown', 'minimum', 'maximum', and 'more' (the latter to indicate that the process of building the result set continues). The client will be required to understand the values 'exact' and 'unknown'. An extensibility mechanism will allow servers to define local values.

5 Progress and Schedule

The Committee released five Committee Drafts in July: See <http://www.loc.gov/standards/sru/oasis.html#cds>.

These include the Abstract Protocol Definition (APD) as well as the SRU 1.2 and OpenSearch bindings. Also included in the release set are the CQL Version 1.2 specification and an Auxiliary Binding for HTTP GET, a companion document to the SRU 1.2 binding.

These will be revised, and an additional three documents will be added:

- Binding for SRU 2.0
- CQL 2.0
- Description Language

The set of eight documents will be submitted for public review in the next six months. Standardization could be completed by late 2009.

6 Conclusions

This work tries to integrate various aspects and approaches to search and retrieval under a unifying model. No single approach is right for every application but if different approaches can be represented in terms of a common abstract model it becomes easier to understand their differences. The OASIS Abstract Protocol Model is intended to serve as the reference model for the different approaches.

SRU and OpenSearch are the two approaches to searching featured by the current OASIS work. These are certainly different approaches. Open Search primarily supports keyword searching, valuable for searching across unstructured documents. The end user often does not know (and does not need to know) how the term is treated by the server. SRU, on the other hand, is useful when the user wants better control over both the query and the results.

By representing both SRU and OpenSearch in terms of a single abstract model, that is, by casting then as bindings of the Abstract Protocol Definition, it becomes easier to compare and contrast the two protocols, and it will be easier to develop software toolkits for implementing both.

SRU and OpenSearch are not the only protocols for search and retrieval; they are the two that the OASIS Committee chose for its focus. We hope that additional protocols will be similarly integrated into this model.

Notes and References

[1] OASIS, the Organization for the Advancement of Structured Information Standards, is a not-for-profit consortium that develops standards for web services, security, and e-business. It was founded in 1993, under the name SGML Open as a consortium promoting SGML interoperability and changed its name in 1998 as its focus changed from SGML to XML. OASIS has more than 5,000 participants representing over 600 organizations and individual members in 100 countries.

[2] SRU is the protocol for Search and Retrieval via URL; in essence, it is Z39.50 adapted to XML and the web. The current version of the SRU specification is at <http://www.loc.gov/sru/>.

[3] CQL, the Contextual Query Language, is the query language developed in conjunction with SRU. Functionally similar to the Z39.50 query language (the Type-1 Query), its design objective is that queries be human readable and writable, and that the language be intuitive while maintaining the expressiveness of more complex languages. The current version of the CQL specifications is at <http://www.loc.gov/standards/sru/specs/cql.html>.

[4] Open Search, a specification developed at Amazon.com covering description documents and other conventions for search engines, is a search protocol that primarily supports keyword searching. It is valuable for searching across unstructured documents, in contrast to SRU, which is more useful for abstract searching and for structured data. The OpenSearch specification (currently draft 3) is at <http://www.opensearch.org/Specifications/OpenSearch/1.1/Draft_3>.

[5] SRW: Search/Retrieve Web Service is described in the SRU Version 1.1 Archive at <http://www.loc.gov/standards/sru/sru1-1archive/srw.html>.

[6] The SRU Via HTTP SOAP specification is at <http://www.loc.gov/standards/sru/specs/transport.html#soap>.

[7] XQuery, "An XML Query Language", is a W3C standard. The current specification, version 1.0, is at: <http://www.w3.org/TR/xquery/>.

[8] The MODS (Metadata Object Description Schema) version 3.3 schema is at: <http://www.loc.gov/standards/mods//v3/mods-3-3.xsd>.