Search Web Services - searchRetrieve Operation: Abstract Protocol Definition Version 1.0

Committee Draft 01

30 June 2008

Specification URIs:
This Version:
http://docs.oasis-open.org/search-ws/june08releases/apd-V1.0-cd-01.doc (Authoritative)
http://docs.oasis-open.org/search-ws/june08releases/apd-V1.0-cd-01.pdf
http://docs.oasis-open.org/search-ws/june08releases/apd-V1.0-cd-01.html

Latest Version:
http://docs.oasis-open.org/search-ws/v1.0/apd-V1.0.doc
http://docs.oasis-open.org/search-ws/v1.0/apd-V1.0.pdf
http://docs.oasis-open.org/search-ws/v1.0/apd-V1.0.html

Technical Committee:
OASIS Search Web Services TC

Chair(s):
Ray Denenberg <rden@loc.gov>
Matthew Dovey <m.dovey@jisc.ac.uk>

Editor(s):
Ray Denenberg rden@loc.gov
Larry Dixson ldix@loc.gov
Matthew Dovey m.dovey@jisc.ac.uk
Janifer Gatenby Janifer.Gatenby@oclc.org
Ralph LeVan levan@oclc.org
Ashley Sanders a.sanders@MANCHESTER.AC.UK
Rob Sanderson azaroth@liverpool.ac.uk

Related work:
This specification is related to:

- Search Retrieve via URL (SRU)

Abstract:
This is an abstract protocol definition for the Search Web Services searchRetrieve operation. It presents the model for the SearchRetrieve operation and is also intended to serve as a guideline for the development of *application protocol bindings.*

Status:
This document was last revised or approved by the OASIS Search Web Services TC on the above date. The level of approval is also listed above. Check the “Latest Version” or “Latest Approved Version” location noted above for possible later revisions of this document.

Technical Committee members should send comments on this specification to the Technical Committee's email list. Others should send comments to the Technical Committee by using the "Send A Comment" button on the Technical Committee's web page at http://www.oasis-open.org/committees/search-ws

For information on whether any patents have been disclosed that may be essential to implementing this specification, and any offers of patent licensing terms, please refer to the Intellectual Property Rights section of the Technical Committee web page (http://www.oasis-open.org/committees/search-ws/ipr.php.

The non-normative errata page for this specification is located at http://www.oasis-open.org/committees/search-ws/.

Notices

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

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

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

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

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

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

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

The names "OASIS", [insert specific trademarked names and abbreviations 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

This document is an abstract protocol definition for the Search Web Services (SWS) searchRetrieve operation. It presents the model for the SearchRetrieve operation and is also intended to serve as a guideline for the development of *application protocol bindings* (hereafter *bindings*, see definitional note).

A binding describes the capabilities and general characteristic of a server or search engine, and how it is to be accessed. A binding may describe a class of servers via a human-readable document (sometimes known as a *profile*, but that term will not be used in this standard); or a binding may be a machine-readable file describing a single server, provided by that server, according to the description language, which is a fundamental component of the SWS standard.

Thus there are two primary types of bindings of interest to this abstract protocol definition: static and dynamic.

- A *static binding* is specified by a human-readable document. A server is known to operate according to that binding at a specific endpoint.
- A *dynamic binding* is a machine-readable description file that the server provides.

There is also a third binding type of interest:

- An *intermediate binding is* specified by a human-readable document, however it binds to one or more dynamic bindings. See Note about Intermediate Bindings. From the point of view of this Abstract Protocol Definition, intermediate bindings are treated as static bindings.

Corresponding to the concepts of static and dynamic bindings, there are two major premises of this standard.

- One premise is that concrete specifications, in the form of static bindings, will be developed and that this abstract protocol definition is to be the foundation for their development, ensuring compatibility among these bindings.
  In this regard it is important to note that this document is not a protocol specification. The static bindings derived from this document are protocol specifications. Examples are SRU 1.1, SRU 2.0, and openSearch.
- Another premise is that any server, even one that existed prior to development of this standard, need only to provide a dynamic binding, that is, a self-description. It need make no other changes in order to be accessible. Furthermore, a client will be able to access any server that provides a description, if only it implements the capability to read the description file and interpret the description, and based on that description to formulate a request (including a query) and interpret the response.

Definitional Note.

In addition to application protocol bindings, there are auxiliary bindings, for example, to bind an application protocol binding to ATOM, or to bind the result to SOAP. However, these auxiliary bindings are not of concern to this abstract protocol definition and are not mentioned further in this document; so this document may refer to application protocol bindings unambiguously as "bindings".

1.1 Terminology

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "NOT RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in **[RFC2119]**. When these words are not capitalized in this document, they are meant in their natural language sense.

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.

2 Abstract Model

This section describes an abstract data model, abstract processing model, and abstract result set model. A binding of this Abstract Protocol Definition should describe its data model, processing model, and result set model in terms of these abstract models.

2.1 Data Model

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 *abstract item* in this model. For purposes of this model there is a single datastore at any given server.

Notes:

- Bindings may use different terminology for various terms:
  - For “abstract item”: “record” or “abstract record”, for example.
  - “datastore”: “database”.
  - “server”:. “search engine”.
- Whenever a binding does use alternative terminology, it should note the alternative usage, referring to the original terminology used in this document.

Associated with a datastore are one or more formats that the server may apply to an abstract item,

Resulting in an exportable structure referred to as a *response Item*.

Note:
the term *item* is often used in this document in place of “abstract item” or “response item” when the meaning is clear from the context or when the distinction is not important.

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

Note:

Bindings may use different terminology for “item type”, for example “schema”.

2.2 Processing Model

A client sends a searchRetrieve request to a server; it responds with a searchRetrieve response. The request includes a search query to be matched against the items at the server’s datastore. The server processes the query, creating a result set (see Result Set Model) of items that match the query. The server may also partition the result set into *result groups*.

Notes:

- Bindings may use different terminology for various terms:
  - “result group”. For example “page”.
  - “searchRetrieve request”. For example “query”. And in turn, that binding would refer to a “query” (as defined in this document) with different terminology, for example “search terms”.

The request also indicates either the desired number of items or which group (by group number) to be included in the response, and includes information about how the individual items in the response, as well as the response at large, are to be formatted.

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

2.3 Result Set Model

This is a logical model; support of result sets is neither assumed nor required by this standard.

There are applications where result sets are critical; on the other hand there are applications where 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.

Processing of a query results in the selection of a set of items, represented by a result set maintained at the server. Logically, it is an ordered list of references to the items. 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 abstract items each referenced by an ordinal number, beginning with 1. The client may request a given item from a result set according to a specific format. For example the client may request item 1 in Dublin Core, and subsequently request item 1 in MODS. The format in which items are supplied is not a property of the result set, nor is it a property of the abstract items as a member of the result set; the result set is simply the ordered list of abstract items.

A server might support requests by item (as in the preceding paragraph) or it may instead support requests by group. It may support one form only or both.

The items in a result set are not necessarily ordered according to any specific or predictable scheme. The server determines the order of the result set, unless it has been created with a request that includes a sort specification. (In that case, only the final sorted result set is considered to exist, even if the server internally creates a temporary result set and then sorts it. The unsorted, temporary result set is not considered to have ever existed, for purposed of this model.) In any case, the order must not change. If a result set is created and subsequently sorted, a new result set must be created and the old result set no longer exists.

Thus, suppose an abstract item is deleted or otherwise becomes unavailable while a result set which references that item still exists. This MUST not cause re-ordering. For example, if a client retrieves items 1 through 3, and subsequently item 2 becomes unavailable, if the server again requests item 3, it must be the same item 3 (see note) that was returned as item 3 in the earlier operation. (If the server requests item 2, and it is no longer available, the server should supply a diagnostic in place of the response item for item 2. Bindings should specify this mechanism in more detail.)

Note:

"Same item" does not necessarily mean the same content; the item's content may have changed.

3 Abstract Parameters and Elements of the SWS searchRetrieve Operation

Abstract request parameters are listed in Table 1 and abstract response elements in Table 2. A binding should list applicable abstract parameters and elements and indicate the corresponding *actual name* of the parameter or element to be transmitted in a request or response.

Note about Intermediate Bindings

> Some bindings are “intermediate bindings”. Similar to static bindings, they are specified in human-readable form, however for intermediate bindings, although the abstract parameters correspond to actual parameters in the binding, the binding is in turn another abstract protocol definition and the actual parameters become abstract parameter to be mapped to the real actual parameters via dynamic bindings. The openSearch binding is an example. For purposes of this Abstract Protocol Definition, these intermediate bindings are treated as static bindings.

The actual name listed in a binding SHOULD be the same as the abstract name, unless there is a reason for it to differ, for example, when a server expects a specific name.

A binding may exclude a particular parameter or element (declare that it is not used). A binding should indicate for every parameter and element used whether it is mandatory or optional, if it is repeatable, and any other usage rules or constraints. A binding may define additional parameters and elements not listed in this abstract protocol definition.

A static binding SHOULD include a table of (or should otherwise list) the request parameters and response elements used in that binding. In addition it should include the following information:

1. **Abstract parameters/elements included**: those defined in the abstract model and included in the binding.
2. **Those Excluded**: those defined in the abstract model and not included in the binding.
3. **Those newly introduced**: those not defined in the abstract model but included in the binding.

3.1 Request Parameters

The Table below shows the abstract parameters of the SWS searchResponse request, including brief descriptions. For more detailed descriptions follow the link provided with the abstract parameter name.

Table 1: Request Parameters

Abstract Parameter Name	Description
responseType	e.g. 'text/html', ‘application/atom+xml’ , application/x+sru
query	The search query of the request.
startPosition	The position within the result set of the first item to be returned.
maximumItems	The number of items requested to be returned.
group	The number of the result group requested to be returned.
responseItemType	e.g. string, jpeg, dc, iso2709. From list provided by server.
sortOrder	The requested order of the result set.

3.2 Response Elements

The Table below shows the abstract elements of the SWS searchResponse response including brief descriptions. For more detailed descriptions follow the link provided with the abstract element name.

Table 2: Response Elements

Abstract Element Name	Description
numberOfItems	The number of items matched by the query.
numberOfGroups	The number of result groups in the result set.
resultSetId	The identifier for the result set created by the query.
item	An individual response item (one of possibly many).
nextPosition	The next position within the result set following the final returned item.
nextGroup	The next result group following the group being returned.
diagnostics	Error message and/or diagnostics.
echoedRequest	The server may echo the request back to the client.

3.3 Parameter and Elements Descriptions

3.3.1 responseType

The **responseType** parameter of the request indicates the type of response to be supplied. This SHOULD be an IANA media/mime type. Examples: 'text/html', 'application/xhtml+xml', 'application/xml', 'application/atom+xml', 'application/x+sru'.

3.3.2 query

The **query** parameter of the request contains a search query to be matched against the datastore at the server creating a result set of items that match the query.

3.3.3 startPosition

The **startPosition** parameter of the request indicates the desired position within the result set of the first item to be returned.

(If the startPosition parameter is included in the request, then the group parameter should not be included.)

For example if the value of this parameter is 2, and the value of the maximumItems parameter is 3, then the request is for items 2, 3, and 4.

Possible values of this parameter are specified in bindings. For example a binding might say that the value must be a positive integer, and that if the request is for the first item within the result set, the value is 1. Another binding might allow the value 'first' or 'last', or 'next'. Default value if this parameter is not supplied and expected server behavior when an invalid value is supplied may be specified by a binding, fixed at a server, or determined by the server for each request.

For example, if the parameter is not supplied, the server might always begin with the next item (following the last item supplied in the previous operation) or might always begin with the first item. If an invalid

value is supplied, for example the value 10 when there are only nine items, the server might not send any items and instead return a diagnostic, or it may begin with the 9th item, or the first item.

3.3.4 maximumItems

The **maximumItems** parameter of the request indicates the number of items requested to be included in the response. Possible values of this parameter are specified in bindings. For example a binding might say that the value must be an integer, and 0 or greater. Another binding might allow the value 'all'. The default value if not supplied may be specified by a binding, fixed at a server, or determined by the server for each request. The server might return less than this number of items, for example if there are fewer matching items than requested, or might declare an error if it cannot return the requested number. The server might return more than this number of items; a binding may indicate that the server will not return more than this number of items, or it may indicate that it might.

3.3.5 Group

The **group** parameter of the request indicates the desired result group to be returned.

(If the group parameter is included in the request, then the startPosition parameter should not be included.)

Possible values of this parameter are specified in bindings. For example a binding might say that the value must be a positive integer, and that if the request is for the first result group within the result set, the value is 1. Another binding might allow the value 'first' or 'last', or 'next'. Default value if this parameter is not supplied and expected server behavior when an invalid value is supplied may be specified by a binding, fixed at a server, or determined by the server for each request.

For example, if the parameter is not supplied, the server might always begin with the next result group (following the last result group supplied in the previous operation) or might always begin with the first result group. If an invalid value is supplied, for example the value 10 when there are only nine groups, the server might not send any group and instead return a diagnostic, or it may send the 9th group, or the first group.

3.3.6 responseItemType

The **responseItemType** parameter of the request indicates the format to be used for the items in the response.

3.3.7 sortOrder

The **sortOrder** parameter of the request indicates the requested order of the result set, for example, which field to sort on, ascending or descending, and so forth.

3.3.8 numberOfItems

The **numberOfItems** element of the response is the number of items matched by the query (the cardinality of the result set). Possible values of this element are specified in bindings. For example a binding might say that the value must be an integer, and 0 or greater. Another binding might list string values with semantics like "unknown" or "too many to count", or a structured value with a number and a confidence level.

3.3.9 numberofGroups

The **numberOfGroups** element of the response is the number of result groups, if the server has partitioned the result set into groups. Possible values of this element are specified in bindings. For example a binding might say that the value must be an integer, and 0 or greater. Another binding might list string values with semantics like "unknown" or "too many to count", or a structured value with a number and a confidence level.

3.3.10 resultSetId

If the server supports result sets, it may include the **resultSetId** element in the response, to be used in a subsequent request, for example to retrieve additional items from the result set, to sort the result set, or to refine the search. (Bindings should specify the mechanism to carry out these functions.)

There will be varying degrees of result set support, for example a server might only support one result set at a time. However the server should attempt to assign a unique name for every result set created so that even when a result sets ceases to exist the client will not mistakenly request items from the new set when meaning to refer to a previous set with the same identifier.

3.3.11 Item

An **item** element of the response (one of possibly many) is one of the items that the server is attempting to return.

3.3.12 nextPosition

The **nextPosition** element of the response indicates the next position within the result set following the final returned item. For example if the result set has six items and the response included items 1 through 4, then the value of this element would be 5.

Possible values of this element are specified in bindings. For example a binding might say that the value must be an integer, and 1 or greater. Another binding might allow string values, for example, ‘end’, indicating that the final returned item was the last. If the result set has six items and the response included items 1 through 6, this might be considered a special case and a binding might declare that the value of nextPosition in this case be 1, or it might specify a special string, for example “done”.

3.3.13 nextGroup

The **nextGroup** element of the response indicates the next result group following the group being returned (meaningful only if the server is responding to a request for a group request rather than a request for items).

Possible values of this element are specified in bindings. For example a binding might say that the value must be an integer, and 1 or greater. Another binding might allow string values, for example, ‘end’, indicating that the final returned item was the last.

3.3.14 diagnostics

The server should supply diagnostics and error messages as appropriate. Bindings should describe relevant details including how diagnostics are to be included and encoded within a response.

3.3.15 echoedRequest

In the **echoedRequest** element of the response, the server may echo the request back to the client along with the response. This is for the benefit of thin clients (such as a web browser) who may not have the facility to remember the query that generated the response it has just received. The manner in which the server encodes the echoed request is specified in bindings.

4 Description and Discovery

A description file SHOULD be provided by a server to describe itself, how it can be queried, and how query results may be interpreted.

Thus there are logically six parts to a description file:

1. General description of the server and its capabilities.
2. How to formulate a request
3. Query grammar
4. How to interpret a response
5. How to Process Results
6. Auto-Discovery Process

When more than one abstract process is defined, the description file may need to include descriptions for each abstract process. At minimum “how to formulate a request” (2) would differ for different abstract processes.

Examples are provided in non-normative Annex B.

A. Acknowledgements

The following individuals have participated in the creation of this specification and are gratefully acknowledged:

Participants:

Kerry Blinco, Australian Department of Education, Employment and Workplace Relations

Ray Denenberg, Library of Congress
Larry Dixson, Library of Congress
Matthew Dovey, JISC
Janifer Gatenby, OCLC/PICS
Ralph LeVan, OCLC
Ashley Sanders, University of Manchester
Rob Sanderson, University of Liverpool

B. Description Language

Non-normative Annex

B.1 Introduction and Background

As noted in the introduction, a binding describes the capabilities and general characteristic of a search engine and how it may be accessed. A binding may be a human-readable document (a static binding), or a machine-readable file (a dynamic binding) provided by that server according to the SWS Description Language, a component of the SWS standard.

A premise of this standard is:

- Any search engine, even one that existed prior to development of this standard, need only provide a self-description. It need make no other changes in order to be accessible.
- A client will be able to access any search engine that provides a description, if only it implements the capability to read the description file and interpret the description, and based on that description to formulate a request (including a query) and interpret the response.

Thus the description language will be a fundamental component of the SWS standard and a major part of the OASIS SWS Technical Committee's work. The description language has not yet been developed. The purpose of this annex is to describe a hypothetical example of a description file.

B.2 Description File Example

The following is a hypothetical description file. It has three sections:

1. General description. Element <databaseInfo>
2. Request formulation. Element <requestInfo>
3. Response interpretation. Element <responseInfo>

```
<sws>
<!-- -->
   <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=cql.any+%3D+%22{query}%22&version=1.1
          &operation=searchRetrieve&maximumRecords={maximumItems}
         &startRecord={startPosition}
     </template>
     <example>
            http://orlabs.oclc.org/SRW/search/scifi
```

```
            ?query=cql.any+%3D+%22ninja+turtles%22&version=1.1
            &operation=searchRetrieve&maximumRecords=10&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>
  </sws>
```

B.3 Description File Components

B.3.1 General Description

The general description component includes general information about the search engine, for example, contact information.

B.3.2 Request formulation

As seen in the example, the request information includes a request template and an example.

The request template includes abstract parameter names enclosed in curly brackets. When valid values for the respective parameters are substituted for the abstract parameter names the result is a valid request.

For example, the template includes:

maximumRecords={maximumItems}

which says in effect that the actual parameter name for the abstract parameter maximumItems is maximumRecords.

B.3.3 Response Interpretation

In the above example an XPath expression (element <tagPath>) is supplied

corresponding to an abstract parameter, indicating where in the response XML that parameter may be found. For example,

```
<item>
    <tagpath>
      /srw:searchRetrieveResponse/srw:records/srw:record/srw:recordData
   </tagpath>
</item>
```

says that the XPath expression to find an element corresponding to the abstract element <item> is:

/srw:searchRetrieveResponse/srw:records/srw:record/srw:recordData