



# searchRetrieve: Part 1. Abstract Protocol Definition Version 1.0

## Candidate OASIS Standard 01

25 October 2012

### Specification URIs

#### This version:

<http://docs.oasis-open.org/search-ws/searchRetrieve/v1.0/cos01/part1-apd/searchRetrieve-v1.0-cos01-part1-apd.doc> (Authoritative)  
<http://docs.oasis-open.org/search-ws/searchRetrieve/v1.0/cos01/part1-apd/searchRetrieve-v1.0-cos01-part1-apd.html>  
<http://docs.oasis-open.org/search-ws/searchRetrieve/v1.0/cos01/part1-apd/searchRetrieve-v1.0-cos01-part1-apd.pdf>

#### Previous version:

N/A

#### Latest version:

<http://docs.oasis-open.org/search-ws/searchRetrieve/v1.0/searchRetrieve-v1.0-part1-apd.doc> (Authoritative)  
<http://docs.oasis-open.org/search-ws/searchRetrieve/v1.0/searchRetrieve-v1.0-part1-apd.html>  
<http://docs.oasis-open.org/search-ws/searchRetrieve/v1.0/searchRetrieve-v1.0-part1-apd.pdf>

#### Technical Committee:

OASIS Search Web Services TC

#### Chairs:

Ray Denenberg ([rden@loc.gov](mailto:rden@loc.gov)), Library of Congress  
Matthew Dovey ([m.dovey@jisc.ac.uk](mailto:m.dovey@jisc.ac.uk)), JISC Executive, University of Bristol

#### Editors:

Ray Denenberg ([rden@loc.gov](mailto:rden@loc.gov)), Library of Congress  
Larry Dixon ([ldix@loc.gov](mailto:ldix@loc.gov)), Library of Congress  
Ralph Levan ([levan@oclc.org](mailto:levan@oclc.org)), OCLC  
Janifer Gatenby ([Janifer.Gatenby@oclc.org](mailto:Janifer.Gatenby@oclc.org)), OCLC  
Tony Hammond ([t.hammond@nature.com](mailto:t.hammond@nature.com)), Nature Publishing Group  
Matthew Dovey ([m.dovey@jisc.ac.uk](mailto:m.dovey@jisc.ac.uk)), JISC Executive, University of Bristol

#### Additional artifacts:

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

- XML schemas: <http://docs.oasis-open.org/search-ws/searchRetrieve/v1.0/cos01/schemas/>
- *searchRetrieve: Part 0. Overview Version 1.0.*  
<http://docs.oasis-open.org/search-ws/searchRetrieve/v1.0/cos01/part0-overview/searchRetrieve-v1.0-cos01-part0-overview.html>
- *searchRetrieve: Part 1. Abstract Protocol Definition Version 1.0.* (this document)  
<http://docs.oasis-open.org/search-ws/searchRetrieve/v1.0/cos01/part1-apd/searchRetrieve-v1.0-cos01-part1-apd.html>
- *searchRetrieve: Part 2. searchRetrieve Operation: APD Binding for SRU 1.2 Version 1.0.*  
<http://docs.oasis-open.org/search-ws/searchRetrieve/v1.0/cos01/part2-sru1.2/searchRetrieve-v1.0-cos01-part2-sru1.2.html>

- *searchRetrieve: Part 3. searchRetrieve Operation: APD Binding for SRU 2.0 Version 1.0.*  
<http://docs.oasis-open.org/search-ws/searchRetrieve/v1.0/cos01/part3-sru2.0/searchRetrieve-v1.0-cos01-part3-sru2.0.html>
- *searchRetrieve: Part 4. APD Binding for OpenSearch Version 1.0.*  
<http://docs.oasis-open.org/search-ws/searchRetrieve/v1.0/cos01/part4-opensearch/searchRetrieve-v1.0-cos01-part4-opensearch.html>
- *searchRetrieve: Part 5. CQL: The Contextual Query Language Version 1.0.*  
<http://docs.oasis-open.org/search-ws/searchRetrieve/v1.0/cos01/part5-cql/searchRetrieve-v1.0-cos01-part5-cql.html>
- *searchRetrieve: Part 6. SRU Scan Operation Version 1.0.*  
<http://docs.oasis-open.org/search-ws/searchRetrieve/v1.0/cos01/part6-scan/searchRetrieve-v1.0-cos01-part6-scan.html>
- *searchRetrieve: Part 7. SRU Explain Operation Version 1.0.*  
<http://docs.oasis-open.org/search-ws/searchRetrieve/v1.0/cos01/part7-explain/searchRetrieve-v1.0-cos01-part7-explain.html>

#### Related work:

This specification is related to:

- Search/Retrieval via URL. The Library of Congress. <http://www.loc.gov/standards/sru/>
- OpenSearch » 1.1 » Draft 5 specification.  
[http://www.opensearch.org/Specifications/OpenSearch/1.1/Draft\\_5](http://www.opensearch.org/Specifications/OpenSearch/1.1/Draft_5)

#### Abstract:

This document is the Abstract Protocol Definition (APD) for searchRetrieve operation, one of a set of documents for the OASIS Search Web Services (SWS) initiative. It presents the model for the SearchRetrieve operation and serves as a guideline for the development of *application protocol bindings* describing the capabilities and general characteristic of a server or search engine, and how it is to be accessed. It defines abstract request parameters and abstract response elements; bindings indicate the corresponding actual names of the parameters and elements to be transmitted in a request or response.

#### Status:

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

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

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

#### Citation format:

When referencing this specification the following citation format should be used:

##### **[SearchRetrievePt1]**

*searchRetrieve: Part 1. Abstract Protocol Definition Version 1.0.* 25 October 2012. Candidate OASIS Standard 01. <http://docs.oasis-open.org/search-ws/searchRetrieve/v1.0/cos01/part1-apd/searchRetrieve-v1.0-cos01-part1-apd.html>.

---

## Notices

Copyright © OASIS Open 2012. All Rights Reserved.

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

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

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

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

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

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

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

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

---

# Table of Contents

1	Introduction.....	5
1.1	Terminology.....	5
1.2	References.....	5
1.3	Namespace.....	5
2	Overview.....	6
2.1	Bindings.....	6
2.2	Abstract Parameters and elements.....	6
3	Abstract Model.....	8
3.1	Data Model.....	8
3.2	Processing Model.....	8
3.3	Result Set Model.....	9
4	Abstract Parameters and Elements of the SWS searchRetrieve Operation.....	10
4.1	Request Parameters.....	10
4.2	Response Elements.....	11
4.3	Parameter and Elements Descriptions.....	11
4.3.1	responseFormat.....	11
4.3.2	query.....	11
4.3.3	startPosition.....	11
4.3.4	maximumItems.....	12
4.3.5	Group.....	12
4.3.6	responseItemType.....	12
4.3.7	sortOrder.....	12
4.3.8	numberOfItems.....	12
4.3.9	numberOfGroups.....	12
4.3.10	resultSetId.....	13
4.3.11	Item.....	13
4.3.12	nextPosition.....	13
4.3.13	nextGroup.....	13
4.3.14	diagnostics.....	13
4.3.15	echoedRequest.....	13
5	Conformance.....	14
Appendix A.	Acknowledgements.....	15
Appendix B.	Description Language.....	16
B.1	Introduction and Background.....	16
B.2	Description and Discovery.....	16
B.3	Description File Example.....	16
B.4	Description File Components.....	17
B.4.1	General Description.....	17
B.4.2	Request formulation.....	17
B.4.3	Response Interpretation.....	17

---

# 1 Introduction

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

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

Most importantly, the APD defines abstract request parameters and abstract response elements; a binding indicates the corresponding actual names of the parameters and elements to be transmitted in a request or response.

This collection of documents includes three binding (see [list](#)).

Included also in this collection is the CQL (*Contextual Query Language*) specification. CQL is a formal query language; SRU requires the use of CQL.

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

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

The documents in the collection of specifications are:

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

## 1.1 Terminology

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in [RFC2119].

## 1.2 References

All references for the set of documents in this collection are supplied in the Overview document:

*searchRetrieve: Part 0. Overview Version 1.0*

<http://docs.oasis-open.org/search-ws/searchRetrieve/v1.0/csd01/part0-overview/searchRetrieve-v1.0-csd01-part0-overview.doc>

## 1.3 Namespace

All XML namespaces for the set of documents in this collection are supplied in the Overview document:

*searchRetrieve: Part 0. Overview Version 1.0*

<http://docs.oasis-open.org/search-ws/searchRetrieve/v1.0/csd01/part0-overview/searchRetrieve-v1.0-csd01-part0-overview.doc>

---

## 2 Overview

### 2.1 Bindings

An application protocol binding (hereafter *binding*, see [definitional note](#)) 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 is not 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.

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”.*

### 2.2 Abstract Parameters and elements

The APD defines abstract request parameters and abstract response elements (see [Abstract Parameters and Elements of the SWS searchRetrieve Operation](#)). Corresponding to these abstract parameter and element names, a binding lists actual names for each of the parameter or element to be transmitted in a request or response.

**Example.**

The APD defines the abstract parameter `'startPosition'` as “The position within the result set of the first item to be returned.” And the SRU bindings refer to that abstract parameter and note that its name, as used in those specifications is `'startRecord'`. Thus the request parameter `'startRecord'` in those bindings represents the abstract parameter `startPosition` in the APD.

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

---

## 89 3 Abstract Model

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

### 93 3.1 Data Model

94 A server exposes a datastore for access by a remote client for purposes of search and retrieval. The  
95 datastore is a collection of units of data. Such a unit is referred to as an *abstract item* in this model.  
96 For purposes of this model there is a single datastore at any given server.

97 Notes:

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

104 Associated with a datastore are one or more formats that the server may apply to an abstract item,  
105 Resulting in an exportable structure referred to as a *response item*.

106 Note:

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

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

113 Note:

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

### 115 3.2 Processing Model

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

120 Notes:

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

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



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

### 131 **3.3 Result Set Model**

132 This is a logical model; result sets may or may not be supported by a given binding.

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

138 When a query is processed, a set of items is selected, and that set is represented by a result set,  
139 maintained at the server. The result set, logically, is an ordered list of references to the items. Once  
140 created, a result set cannot be modified; any operation that would somehow change a result set, instead,  
141 creates a new result set. Each result set is referenced via a unique identifying string, generated by the  
142 server when the result set is created.

143 From the client point of view, the result set is a set of abstract items each referenced by an ordinal  
144 number, beginning with 1. The client may request a given item from a result set according to a specific  
145 format. For example the client may request item 1 in the Dublin Core format, and subsequently request  
146 item 1 in the MODS [7] format. The format in which items are supplied is not a property of the result set,  
147 nor is it a property of the abstract items as a member of the result set; the result set is simply the ordered  
148 list of abstract items.

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

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

157 Thus, suppose an abstract item is deleted or otherwise becomes unavailable while a result set which  
158 references that item still exists. This **MUST** not cause re-ordering. For example, if a client retrieves items  
159 1 through 3, and subsequently item 2 becomes unavailable, if the server again requests item 3, it must be  
160 the same item 3 (see [note](#)) that was returned as item 3 in the earlier operation. (If the server requests  
161 item 2 and it is no longer available, the server should supply a diagnostic in place of the response item for  
162 item 2. Bindings should specify this mechanism in more detail.)

163 Note:

164 “Same item” does not necessarily mean the same content; the item’s content may have changed.

---

## 4 Abstract Parameters and Elements of the SWS searchRetrieve Operation

165

166

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

170 *Note about Intermediate Bindings*

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

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

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

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

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

### 4.1 Request Parameters

191 The Table below shows the abstract parameters of the SWS searchRetrieve request, including brief  
192 descriptions as well as links to more detailed descriptions.

**Table 1: Request Parameters**

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

193 **4.2 Response Elements**

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

**Table 2: Response Elements**

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

196 **4.3 Parameter and Elements Descriptions**

197 **4.3.1 responseFormat**

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

201 **4.3.2 query**

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

204 **4.3.3 startPosition**

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

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

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

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

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

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

#### 220 **4.3.4 maximumItems**

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

#### 229 **4.3.5 Group**

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

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

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

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

#### 243 **4.3.6 responseItem Type**

244 The **responseItem Type** parameter of the request indicates the format to be used for the items in the  
245 response.

#### 246 **4.3.7 sortOrder**

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

#### 249 **4.3.8 numberOfItems**

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

#### 255 **4.3.9 numberOfGroups**

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

### 261 **4.3.10 resultSetId**

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

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

### 269 **4.3.11 Item**

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

### 272 **4.3.12 nextPosition**

273 The **nextPosition** element of the response indicates the next position within the result set following the  
274 final returned item. For example if the result set has six items and the response included items 1 through  
275 4, then the value of this element would be 5.

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

### 281 **4.3.13 nextGroup**

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

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

### 288 **4.3.14 diagnostics**

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

### 291 **4.3.15 echoedRequest**

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

296 Examples are provided in [\(non-normative\) annex "Description Language"](#).

---

## 297 5 Conformance

298 A conformance clause for a specification describes requirements that a product which implements the  
299 specification must meet in order to conform to the specification. It helps a customer of a product which  
300 claims to implements a specification determine whether the product conforms or does not conform to that  
301 specification.

302 This specification prescribes the construction of an application protocol binding. This conformance clause  
303 therefore specifies what a binding must include in order to claim conformance to this specification.

- 304 • Whenever a binding uses terminology that differs from the terminology used in this  
305 specification, it **MUST** note the alternative usage, referring to the original terminology  
306 used in this document.
- 307 • Whenever a binding employs a model that differs from a corresponding model used in  
308 this specification (e.g. data model, processing model, result set model) it **MUST** note the  
309 alternative usage, referring to the original model used in this document.
- 310 • A binding **MUST** list all request parameter and response elements and indicate for each  
311 whether it is mandatory or optional, if it is repeatable, and any other usage rules or  
312 constraints. It should relate every request parameter to an abstract request parameter if  
313 there is one, and every response element to an abstract response element if there is one.  
314 It **MUST** describe any usage specific to that binding which deviates from the usage  
315 described in this document.
- 316 • A binding **SHOULD** list any excluded abstract request parameter and abstract response  
317 element, that is, any such parameter or element listed in this specification that has no  
318 corresponding parameter in the binding.
- 319 • A binding **SHOULD** also include a separate list of request parameters and response  
320 elements that that have no corresponding abstract request parameter or abstract  
321 response element.

322

---

323 **Appendix A. Acknowledgements**

324 Acknowledgements are supplied in the Overview document: *searchRetrieve: Part 0. Overview Version 1.0*

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

327

---

## 328 Appendix B. Description Language

### 329 Non-normative Annex

#### 330 B.1 Introduction and Background

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

335 A premise of this standard is:

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

342 The description language has not yet been developed, and is not part of the initial phase of the work of  
343 the OASIS SWS Technical Committee. It is left for future work. The purpose of this annex is to describe a  
344 hypothetical example of a description file.

#### 345 B.2 Description and Discovery

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

348 Thus there are logically six parts to a description file:

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

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

#### 358 B.3 Description File Example

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

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

363

```
364 <sws>  
365 <!-- -->  
366 <databaseInfo>  
367 <name>Science Fiction Database</name>  
368 <shortName>SciFi</shortName>
```



```

369     <contact>
370         <name>Ralph LeVan</name>
371         <email>levan@oclc.org</email>
372     </contact>
373 </databaseInfo>
374 <!-- -->
375 <requestInfo>
376     <template>
377         http://orlabs.oclc.org/SRW/search/scifi
378         ?query=cql.any+%3D+%22{query}%22&version=1.1
379         &operation=searchRetrieve&maximumRecords={maximumItems}
380         &startRecord={startPosition}
381     </template>
382     <example>
383         http://orlabs.oclc.org/SRW/search/scifi
384         ?query=cql.any+%3D+%22ninja+turtles%22&version=1.1
385         &operation=searchRetrieve&maximumRecords=10&startRecord=1
386     </example>
387 </requestInfo>
388 <!-- -->
389 <responseInfo type='xml' xmlns:srw='http://www.loc.gov/zing/srw/'>
390     <numberOfItems>
391         <tagpath>/srw:searchRetrieveResponse/numberOfRecords</tagpath>
392     </numberOfItems>
393     <item>
394         <tagpath>
395             /srw:searchRetrieveResponse/srw:records/srw:record/srw:recordData
396         </tagpath>
397     </item>
398     <diagnostics>
399         <tagpath>/srw:searchRetrieveResponse/srw:diagnostics</tagpath>
400     </diagnostics>
401 </responseInfo>
402 </sws>

```

## 403 B.4 Description File Components

### 404 B.4.1 General Description

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

### 407 B.4.2 Request formulation

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

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

412 For example, the template includes:

413 **maximumRecords={maximumItems}**

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

### 416 B.4.3 Response Interpretation

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

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

420 For example,

```
421 <item>
422   <tagpath>
423     /srw:searchRetrieveResponse/srw:records/srw:record/srw:recordData
424   </tagpath>
425 </item>
```

426 says that the XPath expression to find an element corresponding to the abstract element <item> is:  
427 /srw:searchRetrieveResponse/srw:records/srw:record/srw:recordData