Related work:

This specification replaces or supercedes:

- 

This specification is related to:

- 

Declared XML Namespace(s):

- [list namespaces here]
- [list namespaces here]

Abstract:

This document defines an standard query syntax for invoking opaque queries implemented by a server. The syntax is agnostic of and decoupled from the underlying implementation for the query within the server. Clients use this syntax to query services in a standard and interoperable manner.

Status:

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

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's procedures with respect to rights in OASIS specifications can be found at 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 implementors or users of this specification, can be obtained from the OASIS Executive Director.

OASIS invites any interested party to bring to its attention any copyrights, patents or patent applications, or other proprietary rights which may cover technology that may be required to implement this specification. Please address the information to the OASIS Executive Director.

Copyright © OASIS® 1993–2007. All Rights Reserved. OASIS trademark, IPR and other policies apply.

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 paragraph are included on all such copies and derivative works. However, this document itself may not be modified in any way, such as by removing the copyright notice or references to OASIS, except as needed for the purpose of developing OASIS specifications, in which case the procedures for copyrights defined in the OASIS Intellectual Property Rights document 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 RIGHTS OR ANY IMPLIED WARRANTIES OF MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE.

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.
1 Table of Contents

1 Table of Contents .................................................................................................................. 4
1 Introduction .......................................................................................................................... 6
  1.1 Why Parameterized Query Interface .............................................................................. 6
  1.2 Comparison With CQL ................................................................................................... 6
  1.3 Terminology .................................................................................................................. 9
  1.4 Conventions ................................................................................................................ 9
  1.5 In Scope ....................................................................................................................... 9
  1.6 Not In Scope .............................................................................................................. 9
2 Abstract Interface .............................................................................................................. 10
  2.1 SearchRequest / Response Operation .......................................................................... 10
  2.2 SearchRequest ........................................................................................................... 10
  2.3 QueryInvocation ........................................................................................................ 11
  2.4 SearchResponse ......................................................................................................... 12
  2.5 Record ....................................................................................................................... 12
  2.6 Exception Response .................................................................................................. 13
3 Canonical Queries ............................................................................................................. 14
  3.1 Universal Query Semantics ......................................................................................... 14
  3.2 CanonicalQuery: FindByld ........................................................................................ 14
    3.2.1 Query Semantics .................................................................................................. 14
  3.3 CanonicalQuery: FindByKeywords ............................................................................ 14
    3.3.1 Query Semantics ................................................................................................ 15
  3.4 CanonicalQuery: CQLQuery ....................................................................................... 15
    3.4.1 Query Semantics ................................................................................................ 15
4 REST Binding .................................................................................................................... 16
  4.1 SearchRequest Binding ............................................................................................... 16
    4.1.1 Search URL Binding ............................................................................................ 16
    4.1.2 Example: Simple Query Invocation .................................................................... 16
    4.1.3 Example: Use of maximumRecords Parameter ............................................... 16
    4.1.4 Example: Use of startRecord Parameter ............................................................. 17
    4.1.5 Example: Use of language Parameter ................................................................ 17
    4.1.6 Example: Use of responseFormat Parameter ..................................................... 17
  4.2 SearchResponse Binding .............................................................................................. 17
    4.2.1 ATOM responseFormat ....................................................................................... 17
      4.2.1.1 SearchResponse to ATOM feed Mapping .................................................... 17
      4.2.1.2 Record to ATOM entry Mapping ................................................................ 18
    4.2.1.3 Example: ATOM responseFormat ................................................................. 19
5 SOAP Binding .................................................................................................................... 21
  6 OpenSearch Binding (Non-Normative). ......................................................................... 22
    6.1 OpenSearch Description Document .......................................................................... 22
    6.2 OpenSearch URL Template ..................................................................................... 23
    6.3 OpenSearch Response Elements ............................................................................. 23
6 Authentication and Access Control (Normative). ........................................................... 24
    7.1 Authentication ........................................................................................................... 24
    7.2 Authorization and Access Control ......................................................................... 24
Illustration Index

Illustration 1: Parametrized Query: Abstract Information Model ..................................................10
Illustration 2: Record Type Hierarchy ...........................................................................................13

Index of Tables

Table 1: Comparison of CQL and Parameterized Query Interface .................................................9
Table 2: SearchRequest Parameters ..............................................................................................11
Table 3: QueryInvocation Parameters ..........................................................................................12
Table 4: SearchResponse Parameters ..........................................................................................12
Table 5: Record Parameters ..........................................................................................................13
Table 6: FindById Query Specification .........................................................................................14
Table 7: FindByKeywords Query Specification .............................................................................15
Table 8: CQLQuery Query Specification ......................................................................................15
Table 9: SearchResponse to ATOM feed Mapping .......................................................................18
Table 10: Record to ATOM Entry Mapping ..................................................................................19
Table 11: Mapping of Response Parameters to Standard OpenSearch Parameters..................23
1 Introduction

This chapter provides an introduction to the rest of this document.

This document normatively defines the parameterized query search interface. It is currently a specification with no standing.

This specification is an evolution of a parameterized query interface whose origins are in the ebXML RegRep (ISO 15000, part 3, 4) standard. It has been written with the intent of contributing it as an input to the OASIS Search Web Services TC.

It is proposed that the current SRU based specification [SWS-SRU] and this specification be aligned and merged with the resulting uniform syntax support both CQL and Parameterized queries. With this intent in mind, this specification has been written to align as best as possible with [SWS-SRU] and uses the same terms whenever possible. Notably, support for CQL query is available via a canonical parameterized query called CQLQuery as defined in section 3.4.

1.1 Why Parameterized Query Interface

The [SWS-SRU] is based on the model where the client specifies a query using an abstract syntax (CQL) and the server builds the concrete query by mapping the abstract CQL to the concrete syntax (e.g. SQL, XQuery etc.).

Some standards such as ebXML RegRep, OGC WRS support a model where a query is implemented in the server complete with all supported predicates. Parts of the query are parameterized using placeholder parameters or variables. The following listing shows a possible implementation of a parameterized query (Opaque to clients):

```
SELECT ro.* from Service ro, Name_ nm, Description d
WHERE
  AND (nm.parent = ro.id AND UPPER ( nm.value ) = UPPER( ''$name'' ) )
  AND (d.parent = ro.id AND UPPER ( d.value ) = UPPER( ''$description'' ) )
```

Listing 1: Example of Parameterized SQL Query Definition

Clients invoke the query by simply identifying the query by its unique identifier and supplying values for a subset of parameters supported by the query:

```
#Following query invoked parameterized query in previous listing
#and only supplies the $name parameter value
GET http://example.com/search?queryId=<someid>&name=japan
```

Listing 2: Example of Parameterized SQL Query Invocation

The server prunes the query to remove any predicate for which parameters were not supplied before executing it. For our example query the server prunes the query predicate for $description parameter:

```
SELECT ro.* from Service ro, Name_ nm, Description d
WHERE
  AND (nm.parent = ro.id AND UPPER ( nm.value ) = UPPER( ''japan'' ) )
```

Listing 3: Example of Parameterized SQL Query Invocation

1.2 Comparison With CQL

While there are some similarities between CQL and parameterized query there several differences. One big difference between CQL and parameterized query is that whereas in Parameterized Query there is a
complete query pre-configured on the server which is pruned down to a smaller query based upon
parameters supplied, in CQL there is no pre-defined query on the server and instead it is built up from the
CQL predicated supplied by the client.

The table below summarizes some of the differences and their relative tradeoffs:
<table>
<thead>
<tr>
<th>Characteristic</th>
<th>CQL</th>
<th>Parameterized Query</th>
<th>Implications</th>
</tr>
</thead>
<tbody>
<tr>
<td>Server may use any underlying query syntax</td>
<td>Yes</td>
<td>Yes</td>
<td>Makes it more difficult to write clients with CQL PQ is simpler than CQL queries</td>
</tr>
<tr>
<td>Server needs to expose an abstract schema to client</td>
<td>Yes</td>
<td>No, the server only needs to declare the queries its supports and the parameters each query supports.</td>
<td></td>
</tr>
<tr>
<td>Client needs to specify query predicates</td>
<td>Yes</td>
<td>No, they are predefined</td>
<td>CQL allows for more ad hoc queries while PQ constrains the query to supported parameters. CQL HTTP GET URLs tend to be long and require encoding. PQ is simpler than CQL queries and its URLs are not much different from OpenSearch URLs in simplicity.</td>
</tr>
<tr>
<td>Client can specify arbitrary joins across indexed records</td>
<td>No</td>
<td>Yes</td>
<td>PQ allows arbitrarily complex queries while hiding the complexity of the query from clients</td>
</tr>
<tr>
<td>Server may implement a specific query using code</td>
<td>No</td>
<td>Yes</td>
<td>PQ allows server complete flexibility in implementing a query as an algorithm in code with query parameters acting as algorithm parameters</td>
</tr>
<tr>
<td>Supports keyword search</td>
<td>No</td>
<td>Yes</td>
<td>PQ supports keyword search as a standard and required query</td>
</tr>
<tr>
<td>Response may include non-XML records in a standard manner</td>
<td>No, requires application specific XML wrapper</td>
<td>Yes, supports both a URL link as well as standard XML wrapper for inline non-XML content</td>
<td>PQ handles non-XML content flexibly and in a standard manner. This is important for many domains and for client interoperability across applications-specific records</td>
</tr>
<tr>
<td>Expects server to maintain state between client requests</td>
<td>Yes</td>
<td>No</td>
<td>PQ is more RESTful</td>
</tr>
<tr>
<td>Supports a RESTful HTTP GET Binding</td>
<td>Somewhat, some areas need minor rework</td>
<td>Yes</td>
<td>PQ is more RESTful</td>
</tr>
<tr>
<td>Supports SOAP Binding</td>
<td>Yes</td>
<td>Yes</td>
<td></td>
</tr>
<tr>
<td>Supports ATOM Binding</td>
<td>No</td>
<td>Yes</td>
<td>ATOM 1.0 provides a standard, extensible syntax for search results. It also makes search</td>
</tr>
</tbody>
</table>
1.3 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 IETF RFC 2119 [RFC2119].

1.4 Conventions

Throughout the document the following conventions are employed to define the data structures used. The following text formatting conventions are used to aide readability:

- UML Diagrams
  UML diagrams are used as a way to concisely describe information models in a standard way. They are not intended to convey any specific Implementation or methodology requirements.

1.5 In Scope

The following functionality is within the scope of this specification.

- Define an abstract interface for clients to invoke opaque queries implemented by a server by simply identifying the query and providing values for its parameters
- Define a REST binding over HTTP transport protocol for parameterized query abstract interface
- Define a SOAP binding for parameterized query abstract interface
- Define OpenSearch profile for discovery and invocation of a parameterized query
- Define ATOM profile for search results of a parameterized query

1.6 Not In Scope

The following functionality is outside the scope of this specification.

- Provide a tutorial on SOAP, REST, ROA, ATOM, OpenSearch, SRU, ebXML RegRep
2 Abstract Interface

This chapter defines the Parameterized Query abstract interface. Later chapters will define various concrete bindings of this interface.

Illustration 1 below shows the the Parameterized Query abstract interface and information model as a UML class diagram. Note that to improve readability, not all attributes are shown in the diagram.

2.1 SearchRequest / Response Operation

This is the only operation defined by the parameterized query interface. A client issues a SearchRequest to a server and the server responds by returning a SearchResponse or a SearchException.

2.2 SearchRequest

A SearchRequest represents a search request initiated by a client to the server. A client MAY issue a SearchRequest with the following parameters.

Illustration 1: Parametrized Query: Abstract Information Model
<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Occurrence</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>id</td>
<td>xsd:string</td>
<td>Optional</td>
<td>If specified, it MUST be a globally unique UUID URN. If not specified the server MUST generate an id and use it to correlate the SearchRequest with it corresponding SearchResponse.</td>
</tr>
<tr>
<td>responseFormat</td>
<td>xsd:string</td>
<td>Optional</td>
<td>The schema in which the response is to be supplied. If this parameter is omitted, the “atom” (ATOM 1.0) response format is assumed (as described in 4.2.1.) Other possible values are ‘rss2.0’, and ‘html’.</td>
</tr>
<tr>
<td>version</td>
<td>xsd:string</td>
<td>Mandatory</td>
<td>The version of the request, and a statement by the client that it wants the response to be less than, or preferably equal to, that version.</td>
</tr>
<tr>
<td>queryInvocation</td>
<td>QueryInvocation</td>
<td>Mandatory</td>
<td>Contains all information that is required to invoke a specific parameterized query. See 2.3 for details.</td>
</tr>
<tr>
<td>startRecord</td>
<td>xsd:integer</td>
<td>Optional</td>
<td>The position within the sequence of matched records of the first record to be returned. The first position in the sequence is 0. The value supplied MUST be greater than or equal to 0. The default value if not supplied (and if records are present in the response) is 0. Same as startIndex in OpenSearch.</td>
</tr>
<tr>
<td>maximumRecords</td>
<td>xsd:integer</td>
<td>Optional</td>
<td>The number of records requested to be returned. Default value if not supplied is determined by the server. The server MAY return less than this number of records, for example if there are fewer matching records than requested, but MUST NOT return more than this number of records. Same as count in OpenSearch. This parameter specifies the language used to represent the search results. The default value is implementation or deployment specific. Same as language in OpenSearch.</td>
</tr>
<tr>
<td>language</td>
<td>xsd:lang</td>
<td>Optional</td>
<td></td>
</tr>
</tbody>
</table>

Table 2: SearchRequest Parameters

2.3 QueryInvocation

A QueryInvocation identifies a parameterized query supported by the server and includes any parameters supported by the parameterized query. It supports the following parameters:
<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Occurrence</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>query</td>
<td>xsd:string</td>
<td>Mandatory</td>
<td>Identifies the parameterized query to be used by the SearchRequest. Must contain the id of a parameterized query.</td>
</tr>
<tr>
<td>queryParameters</td>
<td>Set&lt;QueryParameter&gt;</td>
<td>Optional</td>
<td>The set of query parameters (if any) to be used by the parameterized query for the SearchOperation.</td>
</tr>
</tbody>
</table>

Table 3: QueryInvocation Parameters

### 2.4 SearchResponse

A SearchResponse represents a response sent by server to client in response to a SearchRequest initiated by the client. Upon successful processing of a SearchRequest a server MUST issue a SearchResponse. A SearchResponse is primarily a container for Records matched by the SearchRequest.

A SearchResponse supports the following parameters:

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Occurrence</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>requestId</td>
<td>xsd:string</td>
<td>Mandatory</td>
<td>The id of the original SearchRequest.</td>
</tr>
<tr>
<td>version</td>
<td>xsd:string</td>
<td>Mandatory</td>
<td>The version of the response. This MUST be less than or equal to the version requested by the client in the SearchRequest. See .</td>
</tr>
<tr>
<td>numberOfRecords</td>
<td>xsd:integer</td>
<td>Mandatory</td>
<td>The number of records matched by the query. If the query fails this MUST be 0.</td>
</tr>
<tr>
<td>records</td>
<td>List&lt;Record&gt;</td>
<td>Optional</td>
<td>A sequence of records matched by the query. See 2.5.</td>
</tr>
<tr>
<td>nextRecordPosition</td>
<td>xsd:integer</td>
<td>Optional</td>
<td>The next position within the result set following the final returned record. If there are no remaining records, this field MUST be omitted.</td>
</tr>
</tbody>
</table>

Table 4: SearchResponse Parameters

### 2.5 Record

A Record represents a record matched by the search and returned by server to client within a SearchRequest initiated by the client. A record may represent any type of record including but not limited to XML Records. Illustration 2 shows the type hierarchy defined by this specification for records.
It supports the following parameters:

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Occurrence</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>title</td>
<td>xsd:string</td>
<td>Optional</td>
<td>The title or name for the record. This is analogous to filename for the record.</td>
</tr>
<tr>
<td>summary</td>
<td>xsd:integer</td>
<td>Optional</td>
<td>The summary description for the record.</td>
</tr>
<tr>
<td>updated</td>
<td>xsd:timestamp</td>
<td>Optional</td>
<td>The timestamp for when the record was last updated.</td>
</tr>
<tr>
<td>author</td>
<td>Person</td>
<td>Optional</td>
<td>The person that authored the document</td>
</tr>
</tbody>
</table>

Table 5: Record Parameters

2.6 Exception Response

This section defines the response structure a server MUST return when an exception occurs during the processing of the SearchRequest.

Details TBD but straightforward model that will be mapped to XML element and used within SOAP Faults in SOAP binding.
3 Canonical Queries

This chapter defines a set of parameterized queries that MUST be supported by a server that implements
the parameterized query interface. It also serves as a model for other specifications to specify additional
parameterized queries.

3.1 Universal Query Semantics

This section describes the query semantics that are the same across all queries:

- The client SHALL identify the query using its canonical identifier.
- The client SHALL supply all mandatory parameters.
- The client MAY supply zero or more non-mandatory query parameters when invoking the query.
- A Catalog implementation SHALL prune any predicates of the parameterized query that contain
  parameters for which no value was supplied by the client. A client may narrow or widen the
  specificity of the search by supplying more or less parameters.
- The query SHALL match all remaining predicates for which parameter values have been supplied
  by client.
- A server implementation SHALL silently ignore any parameter values specified by the client for
  any parameter name that is not specified for the query.

3.2 CanonicalQuery: FindById

Query Identifier: urn:oasis:names:tc:search-ws:param-query:query:findById:1.0


<table>
<thead>
<tr>
<th>Parameter Name</th>
<th>Parameter Definition</th>
<th>Occurance</th>
<th>Data Type for Parameter Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>$id</td>
<td>An unique reference to the record within the server</td>
<td>Mandatory</td>
<td>xsd:string that serves as a unique identifier for the record</td>
</tr>
</tbody>
</table>

Table 6: FindById Query Specification

3.2.1 Query Semantics

This query is searches for a record by the identifier specified by $id parameter. If a record is found then it
is returned. If a record is not found then the server MUST return an ObjectNoFoundException error
response.

3.3 CanonicalQuery: FindByKeywords

Query Identifier: urn:oasis:names:tc:search-ws:param-query:query:findByKeywords:1.0

Invocation Example: /search?queryId=urn:oasis:names:tc:search-ws:param-query:query:findByKeywords:1.0&keywords=japan;automobile

<table>
<thead>
<tr>
<th>Parameter Name</th>
<th>Parameter Definition</th>
<th>Occurance</th>
<th>Data Type for Parameter Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>$keywords</td>
<td>A set of keywords that</td>
<td>Mandatory</td>
<td>xsd:string where</td>
</tr>
</tbody>
</table>
are used to perform a search each keyword is separated by a ; character

Table 7: FindByKeywords Query Specification

3.3.1 Query Semantics

This query searches for record that contain one or more of the keywords specified by $keyword parameter. If a record is found then it is returned. If a record is not found then the server MUST return an ObjectNoFoundException error response.

TODO: Need to specify additional parameters to allow AND OR NOT controls

3.4 CanonicalQuery: CQLQuery


<table>
<thead>
<tr>
<th>Parameter Name</th>
<th>Parameter Definition</th>
<th>Occurance</th>
<th>Data Type for Parameter Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>$query</td>
<td>Specifies the CQL expression for the query</td>
<td>Mandatory</td>
<td>xsd:string that MUST conform to the CQL syntax</td>
</tr>
</tbody>
</table>

Table 8: CQLQuery Query Specification

3.4.1 Query Semantics

This query searches for record that match the CQL expression specified by the $query parameter. If a record is found then it is returned. If a record is not found then the server MUST return an ObjectNoFoundException error response.
4 REST Binding

A server MUST support a REST binding to the abstract interface as defined in this section. The REST binding enables a client to invoke parameterized queries using an HTTP GET request.

4.1 SearchRequest Binding

A SearchRequest is bound to an HTTP GET request as defined in this section.

4.1.1 Search URL Binding

The QueryRequest and all its constituent objects are bound to an HTTP URL as defined in this section.

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

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

The rules for binding a SearchRequest to a Search URL structure are as follows:

- The URL to the search algorithm MAY be implementation specific upto and including the `abs_path_prefix`
- The URL MUST have a `abs_path_suffix` of `/search`
- Each primitive attribute of the QueryRequest is mapped as a queryOption with name matching the attribute name and value matching the attribute value
- The query attribute of QueryInvocation is mapped as a queryOption with name “query” and value matching the query attribute value
- Each QueryParameter in the queryParameters attribute of QueryInvocation is mapped as a queryOption with name and value matching the name and value of the QueryParameter

4.1.2 Example: Simple Query Invocation

```
#Template URI for parameterized query invocation
/search/?query={queryId}&[|param-name={param-value}]*
```

Example: Find resource by keywords
```
#query has a `keywords` param which accepts a list of keywords
#separated by ';' character.
/search/?query=urn:oasis:names:tc:ebxml-regrep:query:FindObjectByKeywords&keywords=automobile;japan
```

Listing 4: Example: Simple Query Invocation

The query parameter MUST specify the id of a parameterized query while zero or more additional queryOptions MAY provide parameter name and value pairs for parameters supported by the query or SearchRequest.

4.1.3 Example: Use of maximumRecords Parameter

```
#Example: Find 10 resources by keywords
/search/?queryId=urn:oasis:names:tc:ebxml-regrep:query:FindObjectByKeywords&keywords=automobile;japan&maximumRecords=10
```

Listing 5: Example: Use of maximumRecords Parameter
4.1.4 Example: Use of startRecord Parameter

```plaintext
#Example: Find 10 resources by keywords starting at index 30
/search/?queryId=urn:oasis:names:tc:ebxml-regrep:query:FindObjectByKeywords&keywords=automobile;japan&count=10&startRecord=30
```

Listing 6: Example: Use of startRecord Parameter

4.1.5 Example: Use of language Parameter

```plaintext
#Example: Find resources by keywords using en-us language
/search/?queryId=urn:oasis:names:tc:ebxml-regrep:query:FindObjectByKeywords&keywords=automobile;japan&language=en-us
```

Listing 7: Example: Use of language Parameter

4.1.6 Example: Use of responseFormat Parameter

This canonical parameter specifies the representation format used to represent the search results. The default is XML.

```plaintext
#Example: Find 10 resources by keywords using en-us language and ATOM format
/search/?queryId=urn:oasis:names:tc:ebxml-regrep:query:FindObjectByKeywords&keywords=automobile;japan&language=en-us&responseFormat=atom
```

Listing 8: Example: Use of responseFormat Parameter

4.2 SearchResponse Binding

A SearchResponse is bound to an HTTP GET response as defined in this section.

Upon successful processing of an HTTP GET SearchRequest a server MUST return an HTTP GET response with a status of 200 (OK). The response document format SHOULD be the format specified in the responseFormat attribute of the SearchRequest.

If the server does not support the requested responseFormat then it MUST send an error response with a UnsupportedFormatException document and an appropriate error status code as defined in 2.6.

If the SearchRequest did not specify a responseFormat then the server MUST assume a default value of “atom” responseFormat.

4.2.1 ATOM responseFormat

This section defines a SearchResponse binding to the atome (ATOM 1.0) responseFormat. A server MUST support the atom responseFormat. The atom response format allows search results to be treated as ATOM feeds which may be read by a wide variety of feed readers that support ATOM 1.0.

This document assumes that the reader is familiar with ATOM and is not intended to provide an introduction to ATOM.

4.2.1.1 SearchResponse to ATOM feed Mapping

Table 9 defines the mapping between a SearchResponse and ATOM 1.0 feed syntax using an XPATH syntax to represent objects and attributes within the abstract model and ATOM syntax. Note that any primitive attributes of the SearchResponse not shown in table below are mapped to a <sws:Slot> extension element where each Slot element is simply a holder for a name value pair.
<table>
<thead>
<tr>
<th>SearchResponse Datum</th>
<th>ATOM Mapping</th>
<th>Occurrence</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>/sws:SearchResponse</td>
<td>/atom:feed/</td>
<td>Mandatory</td>
<td>The SearchResponse is mapped to an ATOM feed</td>
</tr>
<tr>
<td></td>
<td>/atom:feed/atom:id</td>
<td>Mandatory</td>
<td>The feed id is the same as the is of the original request</td>
</tr>
<tr>
<td></td>
<td>/atom:feed/atom:title</td>
<td>Mandatory</td>
<td>A server generated title. Servers SHOULD base the title upon the name of the parameterized query</td>
</tr>
<tr>
<td>Alternate Representations available</td>
<td>/atom:feed/atom:link[@rel= &quot;alternate&quot;][@type=&quot;&lt;mime type&gt;&quot;]/@href</td>
<td>Optional, multiple allowed</td>
<td>Links to alternate representations available for the search result</td>
</tr>
<tr>
<td>/sws:SearchResponse/ sws:Record</td>
<td>/atom:feed/atom:entry</td>
<td>Optional, multiples allowed</td>
<td>A Record maps to an entry in the ATOM feed</td>
</tr>
</tbody>
</table>

Table 9: SearchResponse to ATOM feed Mapping

### 4.2.1.2 Record to ATOM entry Mapping

Table 10 defines the mapping between a Record and an ATOM Entry syntax using an XPATH syntax to represent objects and attributes within the abstract model and ATOM syntax:
<table>
<thead>
<tr>
<th>Record Datum</th>
<th>ATOM Mapping</th>
<th>Occurance</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>/sws:Record/@recordIdentifier</td>
<td>/atom:feed/atom:entry/atom:id</td>
<td>Mandatory</td>
<td>recordId maps to Entry Id</td>
</tr>
<tr>
<td>/sws:Record/@title</td>
<td>/atom:feed/atom:entry/atom:title</td>
<td>Mandatory</td>
<td>Entry title. A server SHOULD use a value of “Unknown” if it is unable to provide this information.</td>
</tr>
<tr>
<td>/sws:Record/sws:author/sws:name</td>
<td>/atom:feed/atom:author/atom:name /atom:feed/atom:author/atom:email</td>
<td>Mandatory</td>
<td>The name Person that last edited this record. A server SHOULD use a value of “Unknown” if it is unable to provide this information.</td>
</tr>
<tr>
<td>/sws:Record/@published</td>
<td>/atom:feed/atom:entry/atom:published</td>
<td>Mandatory</td>
<td>The time when Record was first published</td>
</tr>
<tr>
<td>/sws:Record/@updated</td>
<td>/atom:feed/atom:entry/atom:updated</td>
<td>Mandatory</td>
<td>The time when Record was last updated</td>
</tr>
<tr>
<td></td>
<td>/atom:feed/atom:entry/content</td>
<td>Optional</td>
<td>The entry content is the repositoryItem of an ExtrinsicObject entry</td>
</tr>
</tbody>
</table>

Table 10: Record to ATOM Entry Mapping

4.2.1.3 Example: ATOM responseFormat

The following listing shows an example of the ATOM responseFormat:

- The SearchResponse is mapped to a atom:feed
- The Records matched within Searchresponse are mapped to atom:entries
- The example shows how records are linked from the atom:entry rather than contained inline. It could also contain an inline XML or non-XML content which is not shown in this example.

```xml
<?xml version="1.0" encoding="utf-8"?>
<feed xmlns="http://www.w3.org/2005/Atom"
      xmlns:rim="urn:oasis:names:tc:ebxml-regrep:xsd:rim:4.0"
      >
  <title>Records matching keywords: automobile;japan</title>
  <link rel="alternate" type="text/html"
        href="/search/?queryId=urn:oasis:names:tc:ebxml-regrep:query:FindObjectByKeywords&keywords=automobile;japan&language=en-us&responseFormat=html"/>
  <updated>2003-12-13T18:30:02Z</updated>
  <id>urn:oasis:names:tc:ebxml-regrep:query:FindObjectByKeywords:automobile-japan</id>
  <entry>
```
Listing 9: Example: ATOM responseFormat
The REST Binding requires very little in terms of defining a Schema (and of course it has no WSDL).

The SOAP Binding is going to require defining a WSDL and schema for the abstract model. This work is deferred for now as we need to see if we will align this work with current SRU based discussion spec.

One a course is decided the SOAP binding will be quite straightforward a mapping of the abstract interface to WSDL and Xml Schema.
6 OpenSearch Binding (Non-Normative)

This chapter describes the [OpenSearch] binding for Parameterized Query interface. Since OpenSearch is not a standard yet, we make this section non-normative. The intent of this section is to encourage servers to support the OpenSearch as described by this chapter.

6.1 OpenSearch Description Document

In order for a server to be searchable by an OpenSearch client it must allow clients to discover the parameterized queries it supports. A server should expose its supported parameterized queries by declaring them in its OpenSearch Description document.

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

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

Listing 10: OpenSearch Description Document Relative URL

The following listing shows an example of an OpenSearch Description document for a server that supports a Parameterized Query interface with two parameterized queries for the canonical queries defined by this specification. Note that the server is free to declare additional queries defined by other specifications. There is a name associated with the Parameterized Query specification (alias "pq"). Each query has its own unique namespace (aliases "idq" and "kwq") and URL template.

```
<?xml version="1.0" encoding="UTF-8"?>
<OpenSearchDescription
  xmlns="http://a9.com/-/spec/opensearch/1.1/"
  xmlns:idq="urn:oasis:names:tc:search-ws:param-query:query:findById:1.0"
  xmlns:kwq="urn:oasis:names:tc:search-ws:param-query:query:findByKeywords:1.0"  >
  <ShortName>Example Search Engine for Parameterized Query</ShortName>
  <Description>Use this engine to execute any Parameterized Query declared in URL elements below.</Description>
  <Tags>Parameterized Query OASIS</Tags>
  <Contact>admin@example.com</Contact>
  <!--Template URL for FindById query-->
  <Url type="application/atom+xml"
    template="http://example.com/search?
      queryId={pw:queryId}&idq.Id={idq:id}&id={pq:id?}&responseFormat={pq:responseFormat?}&version={pq:version}&startRecord={pq:startRecord?}&maximumRecords={pq:maximumRecords?}&language={pq:language?}"/>
  <!--Template URL for FindByKeywords query-->
  <Url type="application/atom+xml"
    template="http://example.com/search?
      queryId={pw:queryId}&kwq:keywords={kwq:keywords}&id={pq:id?}&responseFormat={pq:responseFormat?}&version={pq:version}&startRecord={pq:startRecord?}&maximumRecords={pq:maximumRecords?}&language={pq:language?}"/>
</OpenSearchDescription
```

search-ws-param-query
Copyright © OASIS Open 2005. All Rights Reserved.
Oct 25, 2007
Page 22 of 27
6.2 OpenSearch URL Template

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

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

\[ http\_URL = "http:\/\//host [ ":" port ] [ abs\_path\_prefix "]/ abs\_path\_suffix [ "?" queryOption ] ] \]

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

- The URL template MAY be implementation specific up to and including the abs_path_prefix
- The URL template MUST have an abs_path_suffix of "search"
- The first set of queryOptions should be query-specific. These should be followed by request-specific queryOptions
- Each query-specific queryOption must be declared and have its template parameters as a qualified name (Qname) within a namespace associated with the query
- Each request-specific queryOption that must be declared
  - If the parameter is equivalent to a standard OpenSearch parameter then it should use a parameter name as an unqualified name as defined in Table 11:

<table>
<thead>
<tr>
<th>Request Parameter</th>
<th>OpenSearch Parameter</th>
</tr>
</thead>
<tbody>
<tr>
<td>maximumRecords</td>
<td>count</td>
</tr>
<tr>
<td>startRecord</td>
<td>startIndex</td>
</tr>
<tr>
<td>language</td>
<td>language</td>
</tr>
</tbody>
</table>

*Table 11: Mapping of Response Parameters to Standard OpenSearch Parameters*

- If the parameter has no equivalent standard OpenSearch parameter then its should declare its template as a qualified name (Qname) within the "pq" namespace defined by this specification
- Each template parameter should indicate if it is optional according to the specification or the option within this specification or another specification

6.3 OpenSearch Response Elements

A server must support the ATOM responseFormat defined in section 4.2.1. By default a the OpenSearch response element returned by the server must be in the ATOM responseFormat. If the search specifies an alternate responseFormat then the server must return a response in that responseFormat. If the server does not support the requested responseFormat then it MUST send an error response with a UnsupportedFormatException document and an appropriate error status code as defined in 2.6.
7 Authentication and Access Control (Normative)

7.1 Authentication

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

A server MAY support single sign-on using an external authentication authority using SAML 2.0 protocols and bindings.

A server MAY support any other

7.2 Authorization and Access Control

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

This chapter provides a [WADL] description for the REST Binding for the REST binding for parametrized query abstract interface. The description is non-normative because WADL is currently not an approved standard. A future version of this document may make the WADL description normative once WADL is approved as an open standard.

TBD...
9 Issues (Non-Normative)

- How can we unify the CQL query syntax and parameterized query syntax leveraging the best of both
10 References

10.1 Normative References

[ATOM] The Atom Syndication Format 1.0
http://atompub.org/2005/07/11/draft-ietf-atompub-format-1.0.html

[XML]

10.2 Informative References

[SRU-SWS] Search Web Services Version 1.0 (Discussion Draft)

[ebRIM] ebXML Registry Information Model version 3.0
http://docs.oasis-open.org/regrep/regrep-rim/v3.0/regrep-rim-3.0-os.pdf

[ebRS] ebXML Registry Services Specification version 3.0
http://docs.oasis-open.org/regrep/regrep-rs/v3.0/regrep-rs-3.0-os.pdf

[RESTBOOK] Restfull Web Services, By Leonard Richardson, Sam Ruby, 2007
http://www.oreilly.com/catalog/9780596529260/

[UML] Unified Modeling Language version 1.5
http://www.omg.org/cgi-bin/apps/doc?formal/03-03-01.pdf

[OpenSearch] Open Search
http://www.opensearch.org Specifications/OpenSearch/1.1

[WADL] Web Application Description Language
https://wadl.dev.java.net/