Intelligence Community and Department of Defense Content Discovery & Retrieval Integrated Project Team

IC/DoD Content Discovery & Retrieval Brokered Search Service Specification for OpenSearch Implementations

V1.0-20101025

REVISION/HISTORY

Doc Revision	Revised By	Revision Date	Revisions
0.1		May 5 2010	Initial draft for subgroup review.
0.2	Dave Lemen		Revised based on comments.
0.3	Dave Lemen	June 19 2010	Revised based on comments, harmonized with SOAP spec.
0.4	Pam Preaseau	August 30 2010	Technical Edits
1.0	Dave Lemen	October 25, 2010	Revisions based on community comments

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

1 Introduction

1.1 Service Overview

An OpenSearch search service may serve as a federated search broker to multiple sources. This specification defines an OpenSearch extension with parameters and response elements that support this functionality.

1.2 Relationship to Other CDR Architecture Elements

The CDR Architecture prescribes an abstract-to-concrete model for the development of architecture elements and guidance for content discovery and retrieval. Each layer or tier of the model is intended to provide key aspects of the overall guidance to achieve the goals and objectives for joint DoD/IC content discovery and retrieval. The following graphic, discussed in detail within the CDR Reference Architecture [CDR-RA], illustrates this model.

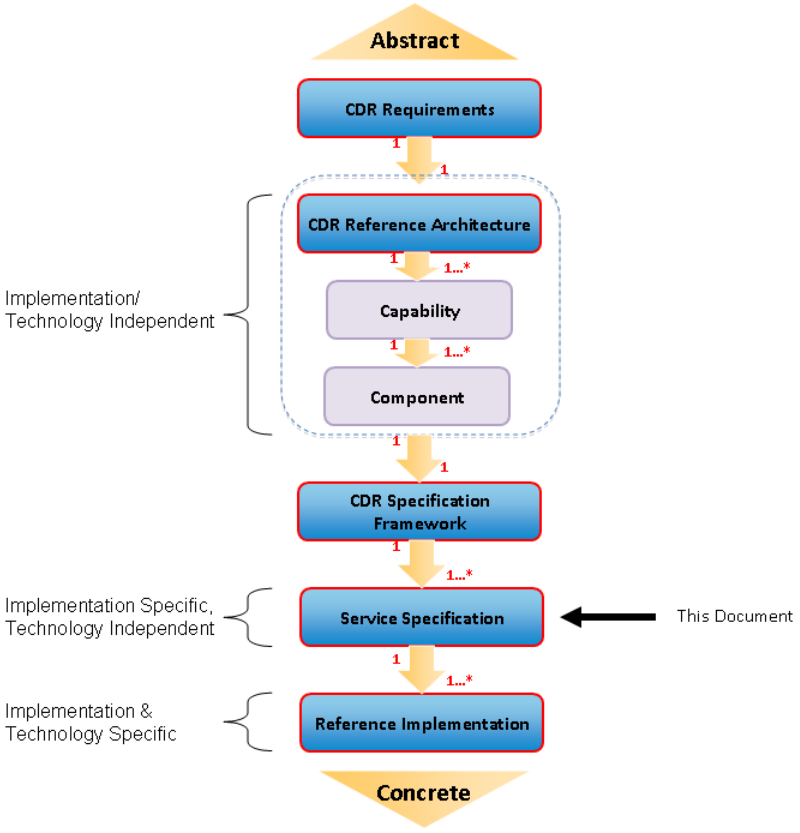


Figure 1 - CDR Architecture Model

As illustrated in Figure 1, the Specification Framework derives from the Reference Architecture (RA) and can describe behavior in terms of the capabilities, components, and usage patterns defined in the RA. The Specification Framework allows multiple

Service Specifications to provide consistent interfaces, both in terms of the structure and semantics of the exchanged information.

This specification provides guidance for implementing the CDR Brokered Search Component using the RESTful OpenSearch [OS] standard. It is intended to provide minimal requirements for implementing an OpenSearch search broker. Additional sub-specifications will provide further guidance for implementation profiles that include specific query types and result formats.

1.3 Notational Convention

The key words "MUST," "MUST NOT," "REQUIRED," "SHALL," "SHALL NOT," "SHOULD," "SHOULD NOT," "RECOMMENDED," "MAY," and "OPTIONAL" in this specification are to be interpreted as described in the IETF RFC 2119. When these words are not capitalized, they are meant in their natural-language sense.

When describing concrete XML schemas and example XML documents, this specification uses XPath as the notational convention. Each member of an XML schema is described using an XPath notation (e.g., /x:RootElement/x:ChildElement/@Attribute).

Examples in this text are distinguished by a black border. These are meant to be illustrative and only one way that the described syntax can be used.

```
<atom:entry>
<atom:title>This is an example.</atom:title>
</atom:entry>
```

1.4 Conformance

Conforming brokered search services MUST support all of the required parameters and elements herein.

1.5 Namespaces

Namespaces referenced in this document and the prefixes used to represent them are listed in the following table.

Table 1 – Referenced XML Namespaces

Prefix	URI	Description
opensearch	http://a9.com/-/spec/opensearch/1.1/	OpenSearch 1.1 (Draft 4)[1]
atom	http://www.w3.org/2005/Atom	Atom 1.0
fs	http://a9.com/-	OpenSearch Federation

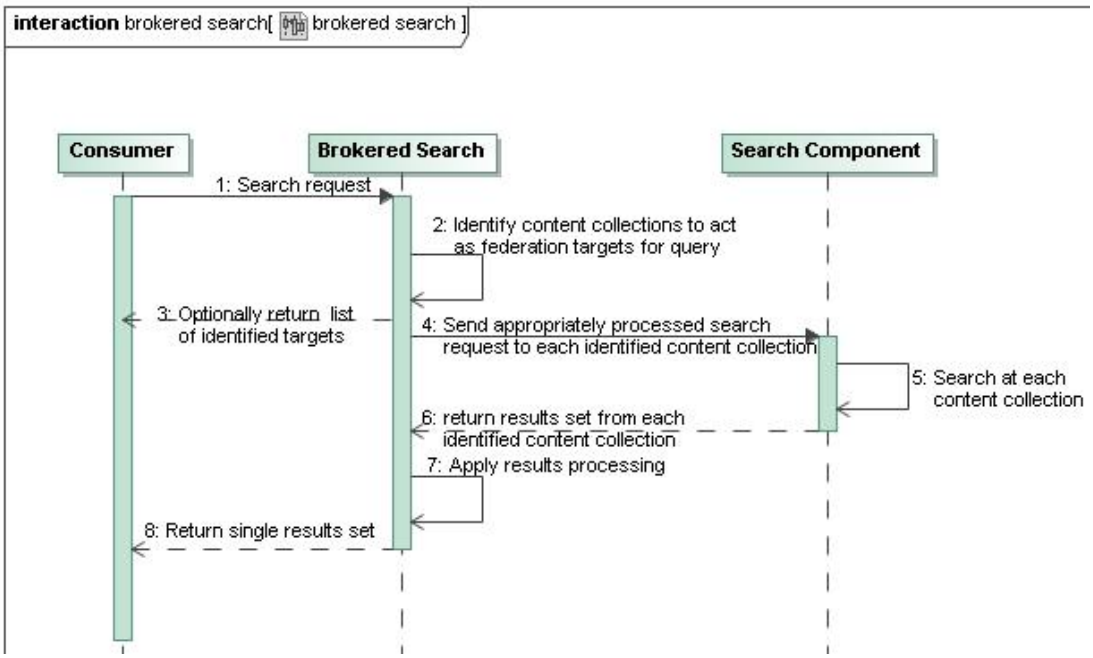
1 *The OpenSearch specification can be found at http://www.opensearch.org/Specifications/OpenSearch/1.1/Draft_4.*

	/opensearch/extensions/federation/1.0/	Extension (proposed)

1.6 License

2 Search Service Behavior

2.1 Main Flow



1. The Consumer invokes the Brokered Search function.
2. The Brokered Search Component identifies the Search Components that will act as the federation targets.
3. The Brokered Search Component may pass a list of identified federation targets to the Consumer.
4. The Brokered Search Component passes the search request to each Search Component identified as a federation target.
5. Search is executed as appropriate at each federation target, generating a results set.
6. Each invoked Search Component returns results set to the Brokered Search Component.
7. The Brokered Search Component carries out the Results Processing activity.
8. The Brokered Search Component returns the processed results to the Consumer, ending the interaction.

2.2 Alternate Flow

2.2.1 Consumer retrieves source list from broker

Prior to step 1, the consumer may request a list of sources from the broker. The broker returns a list of sources with identifiers that the consumer may use to identify the set of sources to which the broker should route a query.

This specification does not support *ad hoc* routing, in which the consumer requests that a query be routed to a back-end source for which the broker does not have a registered endpoint. The process for registering new search components with the broker is outside the scope of this specification.

2.2.2 Broker returns a query identifier

At any point between steps 2 and 8 inclusive, the broker may return a response to the consumer containing a query identifier. The client may use this identifier in subsequent requests for information about the progress and results of that particular query.

2.3 Specification Framework Inputs/Outputs

Table 2 shows each of the parameters and elements defined in this specification, and maps each of them to one or more brokered search activities, as defined in the IC/DoD Content Discovery and Retrieval Specification Framework [CDR-SF] . Items shown in brackets ("[ ]") refer to inputs and outputs defined and described in the CDR-SF (see Table 9 in the CDR-SF).

Table 2. Specification Framework Activities, Inputs, and Outputs

Brokered Search Activity	Inputs	Outputs
Brokered Search Coordination	queryId, [Search Component Inputs]	[Brokered Search Status], [Search Outputs], SourceDescription, shortName, longName, description, link
Source Identification	routeTo	resultSource
Search Component Invocation	maxResults maxTimeout	queryId, sourceStatus, shortName, status, resultsRetrieved, totalResults, elapsedTime
Federation Results Processing	queryId, sourceFilter includeStatus	queryId

The interactions between a brokered search component and its consumer described in this specification use identifiers to track sources (back-end search components) and queries.

Table 3 lists the names of the identifiers for two key tasks, identifying sources and keeping query state, divided into the three locations where they may be used: the OpenSearch Description Document (OSDD), the request parameter, and the response element.

A federated search broker MAY provide stateful interaction in order to improve responsiveness and avoid redundant requests to the back-end sources. The queryId, sourceFilter, and includeStatus parameters allow stateful interaction with the broker for a particular search request.

Table 3. Source and Query Identifiers

Identifier Location	Source Identification	Query State
OSDD Element	sourceDescription/@sourceId	N/A
Request Parameter	routeTo, sourceFilter	queryId
Response Elements	resultSource/@sourceId, sourceStatus/@sourceId	queryId

3 Search Service Interface

3.1 Request Parameters

This extension adds the following request parameters:

- Stateless Initial Request Parameters
  - routeTo
  - maxResults
  - maxTimeout
- Stateful Subsequent Request Parameters
  - queryId
  - sourceFilter
  - includeStatus

3.1.1 The "routeTo" parameter

An-comma-separated list of source identifiers, to which the search query should be routed. The source identifier is found via the "sourceId" attribute in the OpenSearch Description Document (OSDD), described in section 3.2. The same identifier is also referenced by the sourceId attribute in response elements described in section 3.3.

A broker MAY treat the routeTo parameter as optional, by indicating so in the URL templates in its OSDD. That is, if the routeTo parameter is missing, the broker will route the query to a default set of sources or route to a set of sources based on attributes of the query.

There is no significance to the order in which the sources are listed (i.e., it should not be assumed that the sources will be queried in the order they are listed in this parameter). The source identifiers MUST be URL encoded and MUST NOT contain commas.

If a source in the routeTo list is not recognized by the broker, it MUST return an Unknown Source Fault.

Example URL template:

```
http://example.com/?q={searchTerms}&src={fs:routeTo?}
```

Example request:

```
http://example.com/?q=test&src=abc,xyz
```

3.1.2 The "maxResults" parameter

The maximum number of results the federated search broker SHOULD retrieve and process, across all sources. The purpose of this parameter is to improve response times and reduce the amount of storage required of the broker for stateful interactions. The broker MAY determine how to allocate the maxResults number across its back-end sources. For example, it may request the same number of results from each source, or it may vary the number of results requested from the sources. Not all search sources allow a consumer to dictate the maximum number of results, so this parameter may simply serve as a suggestion indicating the consumer's desired maximum number of results.

A broker MAY treat the maxResults parameter as optional, by indicating so in the URL templates in its OSDD.

Example URL template:

```
http://example.com/?q={searchTerms}&src={fs:routeTo?}&mr={fs:maxResults}
```

Example request:

```
http://example.com/?q=test&src=abc,xyz&mr=100
```

3.1.3 The "maxTimeout" parameter

The maximum number of milliseconds the federated search broker should wait for sources to respond. The maxTimeout parameter MAY be passed along to back-end sources that accept a timeout parameter, but it is primarily intended to indicate to the broker how long the consumer is willing to wait for slow-to-respond sources. When passing a maxTimeout value to a back-end source, the broker MAY modify the value to ensure adequate performance.

A broker MAY treat the maxTimeout parameter as optional, by indicating so in the URL templates in its OSDD.

Example URL template:

```
http://example.com/?q={searchTerms}&src={fs:routeTo?}&mt={fs:maxTimeout?}
```

Example request:

```
http://example.com/?q=test&src=abc,xyz&mt=3000
```

3.1.4 The "queryId" parameter

The queryId parameter indicates to the broker which previous request the current request relates to, and it MUST match the string value provided by the broker in response to the initial request in the queryId response element. (See Table 3. Source and Query Identifiers for the names and locations of query identifiers.)

The following example shows the queryId parameter being used to request the second page of results. Parameters that support paging, including the startPage parameter, are defined in the OpenSearch specification [OS].

A broker MAY treat the queryId parameter as optional, by indicating so in the URL templates in its OSDD.

Example of initial request and follow-up request for the second page of results:

Example URL templates:

```
http://example.com/?q={searchTerms}
http://example.com/?id={fs:queryId}&start={startPage?}
```

Response includes:

```
<fs:queryId>1234</fs:queryId>
```

Example request:

```
http://example.com/?id=1234&start=2
```

3.1.5 The "sourceFilter" parameter

The sourceFilter parameter allows the consumer to view results from one particular source. It MUST be used with a queryId parameter, and its value MUST be the URL encoded identifier for one source. A consumer may make multiple requests, each time with a different sourceFilter parameter, to get results for multiple sources.

A broker MAY treat the sourceFilter parameter as optional, by indicating so in the URL templates in its OSDD.

Example URL template:

```
http://example.com/?id={fs:queryId}&filter={fs:sourceFilter}
```

Example request:

```
http://example.com/?id=1234&filter=abc
```

3.1.6 The "includeStatus" parameter

The includeStatus parameter indicates to the broker whether or not the consumer wishes to receive status information on the backend sources with the results. A broker MAY treat the includeStatus parameter as optional, by indicating so in the URL templates in its OSDD. A broker MAY include status information by default and MAY indicate support to requests that contain the queryId for a previous request. A value of "1" indicates that status information should be included in the response (see The "sourceStatus" element). A value of "0" or "" causes sourceStatus not to be included.

Example URL template:

```
http://example.com/?id={fs:queryId}&status={fs:includeStatus?}
```

Example request:

```
http://example.com/?id=1234&status=1
```

Example of initial request and follow-up request for status:

```
http://example.com/?q=test
```

Response includes:

```
<fs:queryId>1234</fs:queryId>
```

Subsequent requests for status on the query:

```
http://example.com/?id=1234&status=1
```

3.2 OpenSearch Description Document Elements

The Web interface for an OpenSearch service is described in an OpenSearch Description document (OSDD). An OpenSearch Brokered Search service may include a list of back-end sources in its OSDD, using the "sourceDescription" element.

Example source XML:

```
<fs:sourceDescription fs:sourceId="abc">
        <fs:shortName>My Source</fs:shortName>
        <fs:longName>My Example Source</fs:longName>
        <fs:description>An OpenSearch service for foo data.</fs:description>
        <fs:link rel="self" type="text/xml" href="http://example.com/description.xml" />
</fs:sourceDescription>
```

3.2.1 The "sourceDescription" element

REQUIRED. This element contains sub-elements describing one back-end source. It contains a REQUIRED attribute, "sourceId". The sourceId attribute value is used to build the list of sources in the "routeTo" request parameter.

3.2.2 The "shortName" element

REQUIRED. Contains a human-readable name identifying the source. MUST be composed of 16 or fewer characters of plain text and MUST NOT contain markup.

3.2.3 The "longName" element

OPTIONAL. Contains a human-readable, extended name identifying the source. MUST be composed of 48 or fewer characters of plain text and MUST NOT contain markup.

3.2.4 The "description" element

OPTIONAL. Contains a human-readable description of the source. MUST be composed of 1024 or fewer characters of plain text and MUST NOT contain markup.

3.2.5 The "link" element

OPTIONAL. Provides access to resources that offer additional information or functionality related to the source. Contains the following attributes:

- href – REQUIRED. Contains a URL for the linked resource.
- rel – REQUIRED. Contains a string representing the relationship of the linked resource. Value may be one of:
  - "self" – The linked resource provides additional information about the source and the content collection(s) it exposes.
- type – REQUIRED. Contains a IANA content type for the linked resource.

3.3 Response Elements

This extension defines three primary response elements, resultSource, queryId, and sourceStatus. The sourceStatus element has a number of child elements associated with it.

3.3.1 The "resultSource" element

REQUIRED. The resultSource element MUST be included with each item returned in the search results, and indicates which back-end source(s) returned it. If de-duplication is applied during processing, multiple sources MAY be associated with the same result.

The resultSource element MUST contain a "sourceId" attribute and the contents of the element MUST be the human-readable short name for the source (see section 3.2.2).

Atom result example:

```
<entry xmlns:fs="http://a9.com/-/opensearch/extensions/federation/1.0/ ">
  <title>This is an Example Page</title>
  <link href="http://example.com/foo/index.html" type="alternate"/>
  <fs:resultSource fs:sourceId="abc">My Source</fs:resultSource>
  <id>http://example.com/foo/index.html</id>
  <date-created>2010-05-05</date-created>
```

```
   <summary>... As the US Army transitions to a force for the 21st Century, so does the
Army&amp;amp;#39;s only independent operational test organization - the US Army
Operational Test ...  </summary>
  </entry>
```

3.3.2 The "queryId" element

OPTIONAL. A broker MAY include a queryId element in its response to indicate that it supports interaction with the cached query result set.

- The queryId element MUST contain a string that identifies a result set cached by the broker.
- The consumer MAY pass the string as the value of the queryId parameter in subsequent requests.
- If the result set referenced by a queryId has been removed from the cache, the broker must return a QueryIdExpired exception.

Atom result example:

```
<feed
        xmlns="http://www.w3.org/2005/Atom"
        os="http://a9.com/-/spec/opensearch/1.1/ "
        xmlns:fs="http://a9.com/-/opensearch/extensions/federation/1.0/ " >

        <title>Search results for "example"</title>
        <fs:queryId></fs:queryId>
…
        <entry>…</entry>
        <entry>…</entry>
…
</feed>
```

3.3.3 The "sourceStatus" element

OPTIONAL. A broker MAY include a sourceStatus element to give the consumer diagnostic information on the overall progress of a search request, and information on each source that was included in the request. If the broker supports the includeStatus parameter (see section 3.1.6), then sourceStatus elements MUST be provided in the response if the consumer requests them by setting fs:includeStatus=1 in the request. When included, there MUST be one sourceStatus element for each source requested. The sourceStatus element MAY be extended with elements from another XML namespace to provide additional information that the broker implementation can provide.

The sourceStatus element MUST contain a "sourceId" attribute.

Example sourceStatus XML:

```
<fs:sourceStatus fs:sourceId="abc">
        <fs:shortName>My Source</fs:shortName>
        <fs:status>waiting</fs:status>
        <fs:resultsRetrieved>100</fs:resultsRetrieved>
```

```
        <fs:totalResults>222222</fs:totalResults>
        <fs:elapsedTime>2000</fs:elapsedTime>
</fs:sourceStatus>
```

3.3.4 The "shortName" element

REQUIRED. This element provides the human-readable short name of the back-end source, consistent with the "shortName child element of the "sourceDescription" element found in the OSDD (see section 3.2.2). The shortName value MAY be the same as the sourceId value. The shortName SHOULD be unique for each source belonging to a particular broker.

3.3.5 The "status" element

REQUIRED. This element reports the current status of a single source. It MUST contain one of the following values:

- excluded – The source was excluded by the broker. There may be a number of reasons for excluding a source, for example, if a maximum number of sources is exceeded, or if the source doesn't support query parameters in the request.
- waiting – The search request has been sent to the source, and the broker is waiting for a complete response from the source.
- error – The source returned an error response.
- timeout – The source failed to respond within the configured timeout period.
- processing – The broker received a complete response from the source, but is processing the result set (e.g., converting format, merging with other results, re-ranking).
- complete – A response was successfully received and the result set from this source has been processed.

3.3.6 The "resultsRetrieved" element

OPTIONAL. This element reports the number of search results that the broker retrieved from the source.

3.3.7 The "totalResults" element

OPTIONAL. This element reports the number of total results matching the query, as reported by the source.

3.3.8 The "elapsedTime" element

OPTIONAL. This element reports the elapsed time, in milliseconds, for the source response.

3.4 Fault Conditions

Fault	HTTP Status	HTTP Status Description

Query Type Not Supported	400	Bad Request
Invalid Query Syntax	400	Bad Request
Query Term Not Supported	400	Bad Request
Query Timeout	500	Server Error
Query Execution Fault	500	Server Error
Query Metadata Fault	400	Bad Request
Security Fault	403	Forbidden
Invalid Paging Value Fault	400	Bad Request
Out Of Range Fault	404	Not Found
Result Sorting Not Supported	400	Bad Request
Result Format Not Supported	406	Not Acceptable
Brokered Search Properties Fault	400	Bad Request
Invocation Results Set Fault	200	OK – Indicate error using sourceStatus element
Invocation Results Optional Output Fault	200	OK – Indicate error using sourceStatus element
Merge Fault	500	Server Error
Unknown Source Fault	400	Bad Request

4 Search Service Implementation

This section provides additional implementation guidance beyond the behavior and interface guidance provided in the previous sections.

4.1 Policy

This specification defines the technical requirements and guidelines for implementing a Brokered Search service. Policy for Brokered Search service implementations is described in auxiliary documents. See the Reference Documents section for a listing of relevant policy documents. Implementers MUST follow the guidance in those policy documents.

4.2 Query Extension Handling

An OpenSearch federated search broker MAY select federation targets based on the query type submitted by the consumer and the supported query types at each back-end source. The query type for a request SHOULD be identified by the presence of OpenSearch extension parameters, such as those belonging to the Geo [OS-GEO] and Time [OS-TIME] extensions, and MAY also be identified by inspecting the contents of the OpenSearch "searchTerms" parameter (e.g., to detect the presence of Boolean operators, AND, OR, NOT, or to detect fielded search terms, such as site:<term>, intitle:<term>, etc.).

A broker SHOULD NOT forward a query to back-end sources that do not support its query type.

4.3 Result Types

The CDR Specification set includes a single predefined Result Type definition that IC/DoD organizations can leverage in their Search service implementations, the IC/DoD Content Discovery and Retrieval Atom 1.0 Result Set Specification [CDR-ATOM]. Implementers SHOULD consult appropriate policy and implementation guidance to determine requirements or recommendations concerning the use of particular Result Types.

4.4 Security Considerations

Any resource may have associated policies for use, especially as applies to authentication and authorization. These policies may be asserted by both the resource owner and those responsible for governance and management of the enterprise. The implementation of policies related to security considerations SHOULD leverage the specific security components and interactions defined by the Joint IC/DoD Security Reference Architecture (SRA), and MUST be in compliance with requirements and guidance for security outcomes as specified in the SRA and its associated specifications.

Implementers using the queryId parameter and element for stateful interactions MUST prevent a requester from using a queryId to access a result set for which he/she is not authorized (e.g., guessing a queryId to gain access to results from a previous requester).

5 Reference Documents

The documents in this section provide the foundation for the Search service. Each document is assigned a reference identifier, which is cited when the document is referenced within this Search Service Specification.

Ref.	Title	Version	Date
CDR-SF	IC/DoD Content Discovery and Retrieval Specification Framework	DRAFT 0.6.1	25 Jan 2010
CDR-RA	IC/DoD Content Discovery and Retrieval Reference Architecture	DRAFT 0.4	16 Dec 2009
ATOM	The Atom Syndication Format http://www.ietf.org/rfc/rfc4287	1.0	Dec 2005
CDR-ATOM	IC/DoD Content Discovery and Retrieval Atom 1.0 Result Set Specification	1.0	March 2010
OS	OpenSearch http://www.opensearch.org/Specifications/OpenSearch/1.1/Draft_4	1.1, Draft 4	2009
OS-GEO	OpenSearch Geo Extension http://www.opensearch.org/Specifications/OpenSearch/	1.0, Draft 1	2009

	Extensions/Geo/1.0/Draft_1		
OS-TIME	OpenSearch Time Extension http://www.opensearch.org/Specifications/OpenSearch/Extensions/Time/1.0/Draft_1	1.0, Draft 1	2010
SRA	Joint IC/DoD Security Reference Architecture	1.0	25 Jul 2008

438