CEOS OpenSearch Developer Guide

Abstract

CEOS has developed a set of conventions around the ATOM syndication specification and OpenSearch 'query-response' specification to enable lightweight mechanisms of data discovery and access (CEOS OpenSearch Best Practices). The result is a data discovery framework that is,

- Lightweight and simple
- Standards-based
- REST-like
- Low entry cost

The purpose of this document is to provide development guidelines to both client and server implementers that achieve the following with respect to any Earth Data OpenSearch implementation,

- Promote the use of the OpenSearch standard as a means of data discovery for Earth Data providers
- Define the expectations and requirements of candidate OpenSearch implementations
- Remove ambiguity in implementation where possible
- Facilitate the aggregation of results between disparate Earth Data providers via OpenSearch common standards
- Allow for clients to access search engines via an OpenSearch Descriptor Document (OSDD) with no a priori knowledge of the interface
- Facilitate smooth integration between related OpenSearch implementations, such as a collection resource that refers to granule resource collections from another provider (International Directory Network (IDN) and the CEOS WGISS Integrated Catalog (CWIC) is a good example)

Table of contents

Introduction

The A9 search technology company, a subsidiary of Amazon, originally developed the OpenSearch specification, which was released to the general community in 2005, and adopted in a variety of settings. The OpenSearch convention is currently maintained on www.opensearch.org, a site provided by A9 to the community.

An OpenSearch Descriptor Document (OSDD) served by an OpenSearch implementation defines a discovery service in terms of,

- HTTP GET query URL, defining simple keyword/value pair query parameters
- Result formats it supports (ATOM, RSS, HTML)

Given that document, a client can execute a query via a simple HTTP GET invocation. The server will respond with a navigable result set containing references to pertinent inventory in terms of

- Spatial extent
- Temporal extent
- Metadata, data and documentation URLs
- Links to further searches (possibly represented by additional OSDDs)

Clients that is as simple as a web browser or as complex as an aggregated portal implementation can use these results.

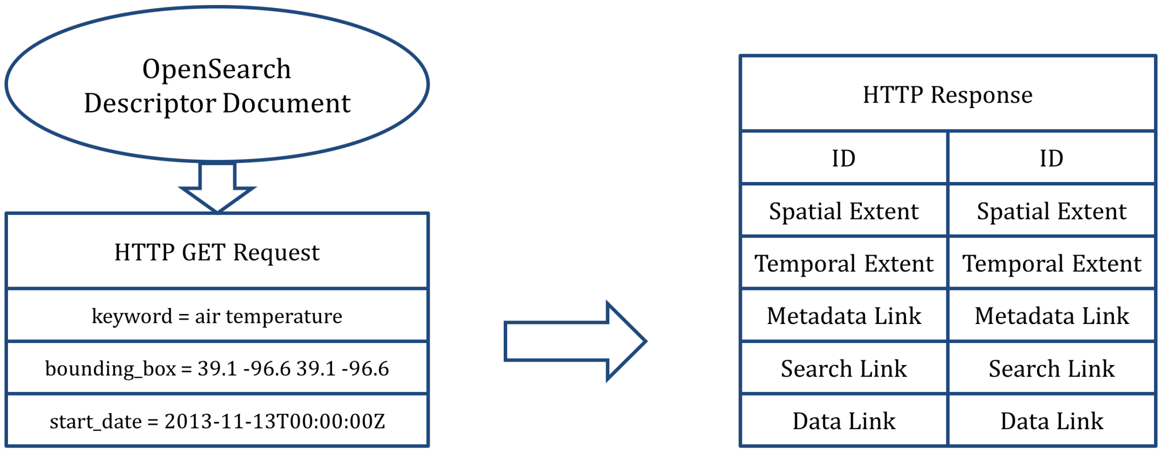

Figure 1: Search Overview

Specification and extension adherence

This guide regards the following specification as mandatory when implementing OpenSearch,

- CEOS OpenSearch Best Practices

The OpenSearch Descriptor Document

The primary purpose of an OpenSearch Descriptor Document (OSDD) is to describe to a client (machine or human) how to search an inventory and what results to expect. Search expectations are defined in terms of protocol, endpoint and query parameters. Result expectations are defined in terms of format and content.

Augmenting an OSDD with parameter elements as per Draft 2 of the parameter extension gives us the ability to

1. Increase the specificity of search parameters.
2. Allow a client, in theory, to programmatically construct a user interface for an arbitrary OpenSearch implementation on-the-fly solely from an OSDD.

Both of these abilities are crucial to one of our goals, the ability to aggregate discovery across multiple, disparate providers.

Anatomy of an OSDD – the server story

They key element of an OSDD is the URL element. Taken as a whole, including any child parameter elements, the entirety of an OpenSearch implementation for a collection of resources can be described to a client.

For example,

```
<Url type="application/atom+xml" type=”collection”
template="http://foo.gov/opensearch/datasets.atom?q={searchTerms?}”/>
```

Tells a client that an inventory of 'dataset' resources can be searched from the endpoint 'foo.gov/opensearch' using the http protocol and that the results can be filtered by the optional query parameter 'q'.
Binding the parameter 'q' to the 'searchTerms' descriptor, where the namespace 'os' refers to the OpenSearch specification namespace, conveys meaning. In this case, the OpenSearch specification defines this meaning as *'keyword or keywords desired by the search client'*

Placeholders for optional search parameters in URL templates must include a question mark character '?' at the end of the search term. To increase interoperability it is recommended that you make as many of your search parameters optional as possible.

The type attribute defines the format of results returned by any query against this resource collection. In this case, the results format is ATOM.

Given the example above the following requests are valid,

```
http://foo.gov/opensearch/datasets.atom

http://foo.gov/opensearch/datasets.atom?q=foo
```

Given this example,

```
<Url type="application/atom+xml"
template="http://foo.gov/opensearch/datasets.atom?q={searchTerms}"/>
```

The following request is NOT valid,

```
http://foo.gov/opensearch/datasets.atom
```

The required parameter "q" and attendant keywords are not included.

OpenSearch server developers should provide an OpenSearch implementation that supports the ATOM results format that may be requested via HTTP content negotiation or resource extension.

Note that in the above examples a 'dataset' resource is inferred by the resource URL. We could formalize this binding by using a relation attribute in the URL element of the OSDD. For example,

```
<Url type="application/atom+xml" rel="collection"
template="http://foo.gov/opensearch/datasets.atom?q={searchTerms}"/>
```

OpenSearch server developers should describe the resource collection endpoint by a relation attribute. The suggested values of the relation attribute can be 'collection' or 'results' (granules).

We can increase the specificity of a query parameter by leveraging the parameter extension. For example, an OpenSearch API that can only return a maximum of 2000 results can be described as follows,

```
<Parameter name="count" value="{count}" minimum="0"
maxInclusive="2000"/>
```

OpenSearch server developers should use 'parameter' elements as per the OpenSearch parameter extension to describe their search-engine-specific search parameters.

A client may wish to generate a user interface at runtime by parsing the OSDD. Leveraging the parameter extension comes close[1] to allowing us to achieve this.

For example, the following parameter definition may instruct a UI to provide a browse radio widget by virtue of the value '{browse}' which describes a Boolean property. That widget would have a display name of 'Browse?' and will be deemed optional by virtue of the minimum value of one.

```
<Parameter name="browse"
      uiDisplay= "Browse required?"
      value="{foo:browse}"
      title="inventory which has a browse image"
      minimum="1" maximum="1">
        <Option value="true" label="Yes"/>
        <Option value="false" label="No"/>
</Parameter>
```

OpenSearch server developers should use name, value, title and uiDisplay[1] attributes for their non-standard[2] parameter definitions.

Some common search parameters have characteristics that are difficult to capture using the parameter extension as it is in Draft 2.

Free text searching, for example, may support query syntax. Given that there is no widely adopted free text query syntax standard and that most implementations will provide this support through 3rd party products like Lucene or Elastic Search, we are proposing an augmentation to the parameter extension that will allow a parameter to exhibit a profile analogous to 'my free text search query language behaves like Lucene'

For example,

```
<Parameter name="keyword"
```

[1] We will petition OpenSearch to add the 'uiDisplay' attribute to the parameter element

[2] By non-standard we mean parameters other than those defined in the OpenSearch specifications and extensions

```
      value="{searchTerms}"
      title="inventory containing all the specified keywords separated
by space, case-insensitive, wildcards are supported"
      minimum="0">
      <link rel="profile"
      href="http://www.elasticsearch.org/guide/en/elasticsearch/referen
      ce/current/query-dsl-query-string-query.html"
      title="This parameter follows the elastic search free text search
      implementations" />
</Parameter>
```

OpenSearch server developers should supply an implementation with free text search capabilities and define the properties of that capability via an ATOM link referring to a profile.

The geometry parameter as defined in the geo extension allows for the representation of a spatial constraint via Well Known Text (WKT). This WKT string may represent a number of different geometries such as point, line or polygon. It is likely that an OpenSearch implementation only supports a subset of these geometries. We are proposing an augmentation to the parameter extension that will allow a parameter to exhibit a number of profiles analogous to 'my geometry search parameters supports point, line and polygon'

For example,

```
<params:Parameter name="geometry"
      value="{geo:geometry}"
      title="inventory which has a spatial extent overlapping this
      geometry"
      minimum="0">
      <atom:link rel="profile"
      href="http://www.opengis.net/wkt/LINESTRING"
      title="This service accepts WKT LineStrings"/>
      <atom:link rel="profile"
      href="http://www.opengis.net/wkt/POINT"
      title="This service accepts WKT Points"/>
      <atom:link rel="profile"
      href="http://www.opengis.net/wkt/POLYGON"
      title="This service accepts WKT Polygons"/>
</params:Parameter>
```

OpenSearch server developers should supply an implementation with geometry search capabilities define an enumeration of supported geometry types via ATOM links referring to geometry type profiles.

You may wish to track client usage through OpenSearch implementations for metrics purposes. You can do this by passing a client ID from implementation to implementation.

OpenSearch server developers should expose an OSDD via the provision of a client-supplied identifier

In the search section of this document we talk about metrics related to an OpenSearch implementation. A key element of metrics support is the ability to identify a client and their usage of your API in some manner. To facilitate this we have introduced the concept of a non-mandatory client id parameter.

In order to minimize the obtrusiveness on the client of a client id we suggest the following,

1. Expose your OSDD via an OpenSearch html landing page.
2. Your OSDD is dynamically generated based on the submission of an html form on that page that has a single, mandatory 'clientId' parameter.
3. Submitting the form will return an OSDD with that client ID embedded in the URL template attribute.

For example invoking,

```
HTTP GET foo.gov/datasets/openSearchDescriptorDocument.xml?clientId=foo
```

Would return an OSDD containing,

```
<Url type="application/atom+xml"
template="http://foo.gov/opensearch/datasets.html?clientId=foo"/>
```

Consequently, if a client abides by the contract of the URL template, all queries to your API will contain the clientId parameter.

Anatomy of an OSDD – the client story

OpenSearch client developers should access an OSDD using an identifier if the server provides such an interface

OpenSearch client developers should leverage as much information provided in the OSDD as possible when interacting with an OpenSearch implementation

The Search Request

An OpenSearch request is a HTTP request (normally 'GET' but this can be defined within an OSDD) directed to a specific collection of resources. Those resources normally describe datasets or granules in earth data discovery.

Anatomy of a Search Request – the server story

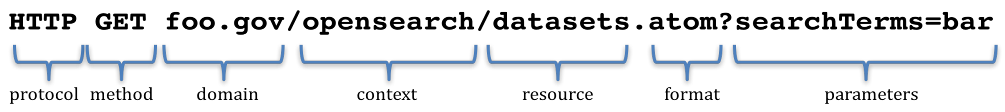

Figure 2: OpenSearch query URL

The set of resources returned can be constrained and traversed by attaching parameters to your request.
The requested format of your results can be defined in the request by content negotiation via the HTTP 'accept' header or using a resource extension.

Example of format request by resource extension,

```
https://foo.gov/opensearch/datasets.atom
```

Example of format request by HTTP 'accept' header,

```
Accept: application/xml+atom: https://foo.gov/opensearch/datasets
```

Query Parameters

We recommend the use of keyword, spatial and temporal constraints for your search implementation. You may define other search parameters but the above 3 have been found to account for over 90% of user queries in a recent study conducted by NASA's ECHO system.

It is expected that multiple query parameters will be AND'd together to form a query.

Search Terms – OpenSearch Specification

The searchTerms parameter is expected to be a simple set of keywords to be used in a free text search. This contrasts somewhat with the traditional method of searching (e.g., within EOSDIS Version 0), which emphasized structured searches based on specific attributes like satellite and instrument. However, free text search is simpler to implement at the client end and more universal, not relying on a common schema, and with judicious use of acronyms by the client or user (e.g., "MODIS") can be nearly as precise. Note that searchTerms are an important discriminator for data collections; however, they are often not useful for files within a data collection, where the discriminator is more likely to be space or time coordinates (see 'two-step searching) of the files. Thus searchTerms is not a required parameter; rather it is the server that specifies in a template whether they are required for that particular search type.

Keyword constraints are key to discovering the appropriate data collections for a user.

Note that if your keyword search algorithm does not conform to any profile (Lucene or Elastic Search for example) then the following is expected:

When multiple searchTerms are included, they should be separated by '+', e.g., "MODIS+fires". Implementations are required to treat this combination as a Boolean "AND" operation by default. You should specify multi-word phrases by setting double quotes (") around the phrase. In this case, an implementation must treat this as a phrase search.

OpenSearch server developers should, if appropriate, filter results by a free text keyword constraint as specified in the OpenSearch specification

Spatial and temporal constraints narrow down a result set to a user's desired area and time of interest.

Spatial – OpenSearch Geo Extension

Bounding box is the simplest and most used type of spatial constraint. As per the geo extension we recommend search providers define a bounding box constraint as 4 longitude/latitude coordinates expressed as EPSG:4326 decimal degrees. The order of those coordinates should be west, south, east and north.

For example,

```
datasets.atom?box=-180.0,-90.0,180.0,90.0
```

The implied relation between a spatial constraint and any inventory returned is that the spatial extent of a resource overlaps the spatial constraint in the query.

Search providers may implement other spatial constraints specified in OpenSearch geo extension version 1.0 draft 2 but support for Geo Bounding Box support is required.

It should be noted that while a bounding box spatial search constraint is expressed in the GEO form, bounding box spatial extents should be expressed as GEORSS bounding boxes (see result section)

OpenSearch server developers should filter results by a geo bounding box constraint as specified in the OpenSearch Geo extension

Temporal – OpenSearch Time Extension

Temporal constraints can be expressed as intervals in time that may be open ended. A upper or lower temporal bound can be expressed as a date or date time constraint specified in the RFC-3339 format as per the OpenSearch Time extension.

Example of a fully specified range,

```
datasets.atom?startTime=2001-01-01T22:00:00Z&endTime=2001-01-
01T22:00:00Z
```

Example of an open-ended range,

```
datasets.atom?startTime=2001-01-01T22:00:00Z
```

Example of an open-ended range with date only,

```
datasets.atom?startTime=2001-01-01
```

The implied relation between a temporal constraint and any inventory returned is that the temporal extent of a resource overlaps the temporal constraint in the query.

OpenSearch server developers should filter results by time start and time end as specified in the OpenSearch Time extension

Result set navigation parameters

We recommend using the index query parameter to navigate through results sets. Any defaults used by your implementation when not supplied by the client should be rendered in your result.

We recommend that a startIndex value of '1' refer to the first result of a result set.

Example,

```
datasets.atom?startIndex=1&count=10
```

Will return the results 1 through 10 of a result set.

```
datasets.atom?startIndex=2&count=10
```

Will return the results 2 through 11 of a result set.

Certain implementations may prefer to use the 'startPage' parameter rather than 'startIndex'. This is acceptable but we recommend the use of 'startIndex'.

Note: the base OpenSearch specification allows startPage as a parameter but does not define startPage in the result body, only startIndex. It is because of this discrepancy that we recommend the use of startIndex. However, we understand that implementations of your search API may preclude the use if index navigation directly.

OpenSearch server developers should allow a client to navigate a result set using the 'startIndex' and 'count' parameters as specified in the OpenSearch specification

Anatomy of a Search Request – the client story

OpenSearch client developers should preserve any non-variable search parameters (such as client id) given to them by the OSDD for any searches related to that OSDD

Metrics parameters

In the OSDD section we talk about the need for the user to supply a client ID when obtaining an OSDD. The OSDD returned will embed that client ID in the URL template. In theory, all requests sent by that client will have that parameter present.

The purpose of this ID is to allow the tracking of metrics based on particular clients and users.

To facilitate this need we recommend that search providers implementing OpenSearch use this client ID reference when logging or generating their own metrics. Furthermore, we recommend that the client ID be propagated to other searches (see 'Two step searching). We strongly recommend that the client ID should be a mandatory parameter when obtaining an OSDD through your API but optional in any search request.

OpenSearch server developers should allow a client to specify a 'clientId' and that ID must be propagated to any secondary OpenSearch implementations

While this is not an ideal means of providing user information, it appears to be the best way to provide 'opt in', non-obtrusive support.
The supplied client ID may be (a) non-unique and (b) not meaningful. This risk is balanced by the possible lack of interest in a provider's OpenSearch API if a more thorough handshake (designated IDs for example) is required.

Example,

```
datasets.atom?clientId=foo
```

Full Example of a 'developer guide' compliant OpenSearch Request

```
https://foo.gov/opensearch/datasets.atom?clientId=foo&searchTerms=air+t
emperature&box=10,10,10,10&startTime=startTime=2001-01-
01T22:00:00Z&endTime=2001-01-01T22:00:00Z&pageNumber=1&count=10
```

The Search Response

An OpenSearch response is an ATOM feed with zero or more ATOM entries. Each entry represents a single resource pertaining to the query submitted.

Anatomy of a response – the server story

An ATOM response is one ATOM feed element containing the following,

1. Information about the search implementation in terms of title, author and ID.
2. Information about the nature of the result set in terms of total number of results, number of results returned and how many the client asked for.
3. Navigation information for traversing that result set including links to the previous, next, first and last results in the set.
4. Zero or more entries pertaining to resources matching the client query

HTTP Response				
ATOM Feed				
Atom Entry 1	Atom Entry 2	Atom Entry 3	Atom Entry 4	Atom Entry 5
Metadata Link	Metadata Link	Metadata Link	Metadata Link	Metadata Link
Search Link	Search Link	Search Link	Search Link	Search Link
Data Link	Data Link	Data Link	Data Link	Data Link
Documentation Link	Documentation Link	Documentation Link	Documentation Link	Documentation Link

Figure 3: OpenSearch Response overview

Result set navigation by content

As described in the request section,

OpenSearch server developers should allow a client to navigate a result set using the 'startIndex'/'startPage' and 'count' parameters as specified in the OpenSearch specification

This recommendation has an impact on how a provider describes their result set. For example,

```
<feed>
        …
        <os:totalResults>3415</os:totalResults>
        <os:itemsPerPage>10</os:itemsPerPage>
        <os:startIndex>1</os:startIndex>
        …
</feed>
```

Indicates that the client asked for the first 10 results where there are 10 results per page and there are a total of 3415 results. The first 10 entries (1-10) would be represented in the feed. If the client asked for the second index onwards (startIndex=2) then the entries 2 to 11 would be represented in the feed.

In the case where zero results are returned the following would be contained in the feed,

```
 <feed>
        …
        <os:totalResults>0</os:totalResults
        …
        <subtitle type="text">Your search yielded zero
matches</subtitle>

</feed>
```

OpenSearch server developers should not include 'itemsPerPage' and 'startindex' in a feed result when zero entries are returned for a query and that the feed subtitle should contain human-readable text describing that state.

Result set navigation by HATEOAS

In order to facilitate traversal of a result set we can leverage the REST concept of Hypermedia as the Engine of Application State (HATEOAS) and provide links to the current, previous, next, first and last pages in the result set as follows,

```
<feed>
        …
        <os:totalResults>55</os:totalResults>
        <os:itemsPerPage>10</os:itemsPerPage>
        <os:startIndex>30</os:startIndex>
        <link
href="foo.gov/opensearch/datasets.atom?startIndex=1&numberOfResults=10"
rel="first" type="application/atom+xml"/>
        <link href="foo.gov/opensearch/datasets.atom?
startIndex=20&numberOfResults=10" rel="prev"
type="application/atom+xml"/>
        <link href="foo.gov/opensearch/datasets.atom?
startIndex=30&numberOfResults=10" rel="self"
type="application/atom+xml"/>
        <link href="foo.gov/opensearch/datasets.atom?
startIndex=40&numberOfResults=10" rel="next"
type="application/atom+xml"/>
        <link href="foo.gov/opensearch/datasets.atom?
startIndex=50&numberOfResults=10" rel="last"
type="application/atom+xml"/>
        …
</feed>
```

OpenSearch server developers should provide navigation links for the first, previous, current, next and last pages of a result set.

Result metadata

Each resource in a result will be represented by an ATOM 'entry' element. That element will contain an ID, a link to the metadata from which this entry was derived, and, if applicable, a spatial extent and a temporal extent.

OpenSearch server developers should provide a link of relation 'alternate' for each entry in a result set feed. That link should point to the original metadata from which this entry was derived.

When rendering a spatial extent for an entry we recommend that an implementation provide a minimum-bounding rectangle (MBR) to represent more complex geometries such as polygon in addition to the accurate spatial extent. The rationale behind this is that bounding rectangle is the lowest common denominator of spatial constraint/extent supported by providers and clients. You may also render your original extent in WKT as per the geo extension.

For example, an MBR representing a point and its original representation,

```
        <georss:box>39.1 -96.6 39.1 -96.6</georss:box>
        <georss:point>39.1 -96.6</georss:point>
```

OpenSearch server developers should render spatial extents using a minimum-bounding rectangle in the 'georss' format as well as the native representation of that extent.

When rendering a temporal extent, an implementation should use a Dublin Core date element, which will support date and date-time ranges, open-ended date and date-time ranges and single dated and date-times.

For example a date-time range,

```
<dc:date xmlns:dc="http://purl.org/dc/elements/1.1/">1984-12-
25T00:00:00.000Z/1988-03-04T00:00:00.000Z</dc:date>
```

an open-ended date-time range,

```
<dc:date xmlns:dc="http://purl.org/dc/elements/1.1/">1984-12-
25T00:00:00.000Z/</dc:date>
```

a single date-time,

```
<dc:date xmlns:dc="http://purl.org/dc/elements/1.1/">1984-12-
25T00:00:00.000Z</dc:date>
```

a single date,

```
<dc:date xmlns:dc="http://purl.org/dc/elements/1.1/">1984-12-
25</dc:date>
```

OpenSearch server developers should render temporal extents using a Dublin Core date element.

Entries should also render link elements pointing to various artifacts associated with the resource. Those may include

- Data
- Additional metadata
- Browse imagery
- Documentation.

OpenSearch server developers should use the following relation values when describing artifacts associated with a resource,

Artifact	Definition	Relation	Example
Metadata	file with (usually) structured information about corresponding data files	via	<link href="foo.xml" rel="via"/>
Browse	image of the data typically used for making data request decisions	icon	<link href="sample.gif" rel="icon"/>
Documentation	File with human-readable information about the resource	describedBy	<link href="foo.pdf" rel="describedBy"/>
Data	link representing a data file or other science data resource; may be large in size	enclosure	<link href="foo.hdf" rel="enclosure"/>
OSDD	link to an OpenSearch Description Document; useful for recursive searching	search	<link href="osdd.xml" rel="search"/>

Table 1: Link relations for OpenSearch response artifacts

Anatomy of a response – the client story

Result set navigation

We recommend that server implementations use HATEOS to provide result set navigation. Client implementations are at the mercy of the implementations they interact with. When supplied with both mechanisms we advocate using HATEOS as it is a more RESTful solution and probably more simple to implement on the client side since the request is already formulated in the response.

Result metadata

The purpose of result metadata with respect to client implementation is to provide the user with information to help determine data of interest. Consequently, a client should provide the following:

- A means of rendering canonical metadata associated with a result entry given a URL
- A means of rendering browse imagery associated with a result entry given a URL
- A means of rendering documentation associated with a result entry given a URL
- A means of accessing data associated with a result entry given a URL
- A means of 'drilling down' on a dataset search for granules (or further depending on server implementation) (see 'Two step searching')

This can be achieved by leveraging the rel type of each link presented in a result entry as per Table 1.

Result ordering – the server story

Result ordering is optional as per OGC recommendations. We recommend that if you support search by free text that you employ relevancy ranking as per the OpenSearch Relevancy Extension and order your results by that ranking in descending order.

For example,
```
<feed xmlns:relevance="http://a9.com/-
/opensearch/extensions/relevance/1.0/">
        <entry>
                …
                <relevance:score>0.98</relevance:score>
                ..
        </entry>
        <entry>
                …
                <relevance:score>0.75</relevance:score>
                ..
        </entry>
        …
</feed>
```

Result ordering – the client story

When portraying results from a service implementation that support relevancy and ordering the client may want to leverage the relevancy scores:

- Displaying the relevancy score
- Aggregating more than one result source and ordering, assuming the two sources abide by the same relevancy rules.

Handling errors – the server story

In the event of an error the service implementation should respond with the appropriate HTTP error code and a response body that verbosely describes the error in ATOM format.
These errors will generally fall into two categories,

- User input errors will be responded to with a BAD_REQUEST 400 response or another error in the 4xx range, if the error corresponds to one of the existing HTTP error codes.
- Processing errors will be responded to with an INTERNAL_SERVER_ERROR 500 response or another error in the 5xx range, if the error corresponds to one of the existing HTTP error codes.

Note that a query that yields zero results is not treated as an error state and, therefore, should return a successful HTTP code.

Two-step Searching

One serious hurdle to overcome in searching for data is the enormous number of data items to account for in responses, as well as the expected number of successful "hits" for a query. In ordinary web searches, the end user is usually looking for a small number of web pages or documents. Relevance ranking typically does a good job of presenting these successful hits near the top of the returned list, followed by single point-and-click retrievals. However, when searching for Earth science data covering large time periods or spatial areas, a user will often specify a set of constraints to find an appropriate data collection together with space-time criteria for granules within that data collection. Often, the precision of the data collections returned for the search is low, with many spurious hits. However, the space-time precision of the files is often quite high: that is, the user truly wants to use all the data granules of a desirable data collection set that fall within the space-time region of interest. Thus, searching for all data satisfying both dataset content and space-time region at the same time can produce a great many spurious hits, i.e., all the granules for data collections that are *not* desired.

The two-step solution breaks a search into two distinct parts. A client will first identify datasets of interest (step 1) and then search for granules associated with those datasets (step 2).

The server story

To facilitate this, when an OpenSearch implementation representing dataset/collection resources returns results, each entry in that result set should contain an ATOM link of relation type 'search' if applicable. That ATOM link should point to a URL of an OSDD pertaining to granules associated with that entry (i.e. dataset/collection).

For example,

```
<feed … >
   …
   <entry … >
    <id>http://foo.gov/opensearch/datasets/foo</id>
         <link rel='search'
type='application/opensearchdescription+xml'>https://datacenter.nasa
.gov/opensearch/granules/descriptorDocument?datasetId=foo&clientId=d
</link>
         …
   </entry>
   …
</feed>
```

This relationship is formalized by two attributes in the link element:

- rel='search'
- type='application/opensearchdescription+xml'

Although two steps, dataset and granule, are illustrated here, the notion of more steps could be used. That is, one could have an OpenSearch server that searches top-level (search engine) documents, returning the URLs to OSDDs for each search engine.
Each step in the process returns links to OpenSearch Description documents, which in turn inform the client on how to execute the next level search, until the lowest level, where the return consists of links to actual data of interest.

The two-step search concept is an optional aspect of the convention. There are use cases where only a dataset query is needed, as well as some where only the granule-level query is needed.

The client story

A client can parse an OSDD and formulate a search for granules belonging to that dataset.

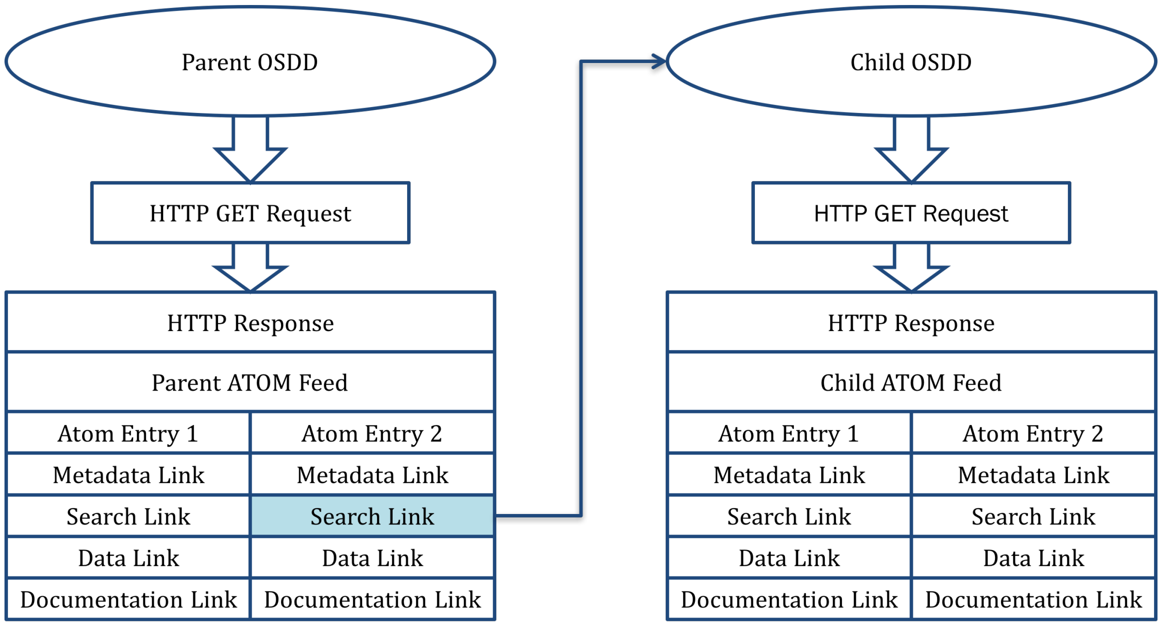

Figure 4: Two-step overview

For example,

1. Given an dataset OSDD, search for datasets of interest

```
GET http://foo.gov/opensearch/datasets?searchTerms=MODIS&clientId=d
```

2. Parse search results

```
<feed … >
   …
   <entry … >
    <id>http://foo.gov/opensearch/datasets/MODIS_dataset</id>
         <link rel='search'
type='application/opensearchdescription+xml'>bar.gov/opensearch/gran
ules/descriptorDocument?datasetId=MODIS_dataset&clientId=d</link>
         …
   </entry>
   …
</feed>
```

3. Retrieve granule OSDD
4. Parse granule OSDD

```
< OpenSearchDescription>
     …
     <Url type="application/atom+xml"
          template="bar.gov/opensearch/granules.atom?datasetId=
MODIS_dataset&clientId=d&box={geo:box?}">
            …
     </Url>
     …
```

```
</OpenSearchDescription>
```

5. Search for granules of interest

```
GET http://bar.gov/opensearch/granules.atom?datasetId=
MODIS_dataset&clientId=d&box=1,2,3,4
```

Appendix A – Summary Table

Element or Attribute	Description	Namespace	Example
title element	Human-readable text describing about an entry, usually relatively short	ATOM	
id element	Unique identifier for a returned resource in ATOM response	ATOM	
link element	URL for a resource referenced in ATOM response	ATOM	
rel attribute	Relationship represented by a given <link> element.	ATOM	rel=”enclosure” (used to describe <link> elements pointing to data resources)
type attribute	Type of resource, typically given as a mime-type	ATOM	type=”application/x-netcdf” (describes link elements that are netCDF files).
Url	External resource location	OpenSearch	<Url type="text/html" template="http://example.com/search?q={searchTerms}&amp;pw={startPage?}" />
template attribute	template attribute of <Url> element Required in OSDD	OpenSearch	
date	Temporal coverage of data in an ATOM response.	Dublin Core	
geo:box	Minimum bounding rectangle, required in ATOM response if spatial information is	OpenSearch Geo	

	included		
geo:geometry	Optional Well-Known-Text based geometry	OpenSearch Geo	

Table 2: summary of common elements and attributes

Appendix B – Examples

OSDD with parameter extension

```
<?xml version="1.0"?>
<OpenSearchDescription
      xmlns="http://a9.com/-/spec/opensearch/1.1/"
      xmlns:echo="http://www.echo.nasa.gov/esip"
      xmlns:geo="http://a9.com/-/opensearch/extensions/geo/1.0/"
      xmlns:time="http://a9.com/-/opensearch/extensions/time/1.0/"
      xmlns:esipdiscovery="http://commons.esipfed.org/ns/discovery/1.2/
" esipdiscovery:version="1.2"
      xmlns:params="http://a9.com/-
/spec/opensearch/extensions/parameters/1.0/"
      xmlns:atom="http://www.w3.org/2005/Atom" >
      <ShortName>ECHO Dataset Search</ShortName>
      <Description>NASA ECHO Dataset search using geo, time and
parameter extensions</Description>
      <Tags>ECHO NASA CWIC CEOS ESIP OGC dataset</Tags>
      <Contact>support@echo.nasa.gov</Contact>
      <Url
            type="application/atom+xml"
            params:method="GET"
            template="https://api.echo.nasa.gov:443/opensearch/datasets
            .atom?keyword={searchTerms?}&amp;instrument={echo:instrumen
            t?}&amp;satellite={echo:satellite?}&amp;boundingBox={geo:bo
            x?}&amp;geometry={geo:geometry?}&amp;placeName={geo:name?}&
            amp;startTime={time:start?}&amp;endTime={time:end?}&amp;cur
            sor={startPage?}&amp;numberOfResults={count?}&amp;uid={geo:
            uid?}&amp;clientId=foo">
            <params:Parameter
                  name="keyword"
                  value="{searchTerms}"
                  minimum="0">
                  <atom:link
                        rel="profile"
                        href="http://www.elasticsearch.org/guide/en/ela
                        sticsearch/reference/current/query-dsl-query-
                        string-query.html"
                        title="This parameter follows the elastic
                        search free text search implementations" />
            </params:Parameter>
            <params:Parameter
                  name="instrument"
                  value="{echo:instrument}"
                  title="Inventory associated with a satellite
                  instrument expressed by this short name"
                  minimum="0"/>
```

```
<params:Parameter
      name="satellite"
      value="{echo:satellite}"
      title="Inventory associated with a Satellite/platform
      expressed by this short name" minimum="0"/>
<params:Parameter
      name="boundingBox"
      value="{geo:box}"
      title="Inventory with a spatial extent overlapping
      this bounding box"
      minimum="0"/>
<params:Parameter
      name="geometry"
      value="{geo:geometry}"
      title="Inventory with a spatial extent overlapping
      this geometry"
      minimum="0">
      <atom:link
            rel="profile"
            href="http://www.opengis.net/wkt/LINESTRING"
            title="This service accepts WKT LineStrings"/>
      <atom:link
            rel="profile"
            href="http://www.opengis.net/wkt/POINT"
            title="This service accepts WKT Points"/>
      <atom:link
            rel="profile"
            href="http://www.opengis.net/wkt/POLYGON"
            title="This service accepts WKT Polygons"/>
</params:Parameter>
<params:Parameter
      name="placeName"
      value="{geo:name}"
      title="Inventory with a spatial location described by
      this name" minimum="0"/>
<params:Parameter
      name="startTime"
      value="{time:start}"
      title="Inventory with a temporal extent containing
      this start time" minimum="0"/>
<params:Parameter
      name="endTime"
      value="{time:end}"
      title="Inventory with a temporal extent containing
      this end time" minimum="0"/>
<params:Parameter
      name="cursor"
      value="{startPage}"
      minimum="0"/>
<params:Parameter
      name="numberOfResults"
      value="{count}"
      minimum="0"
      maxInclusive="2000"/>
<params:Parameter
      name="uid"
      value="{geo:uid}"
```

```
                  title="Inventory associated with this unique ID"
                  minimum="0"/>
      </Url>
      <Query
            role="example"
            echo:instrument="AMSR-E"
            echo:satellite="Aqua"
            title="Sample search"
            geo:box="-180.0,-90.0,180.0,90.0"
            time:start="2002-05-04T00:00:00-0400"
            time:stop="2009-05-04T00:00:00-0400"/>
      <Attribution>NASA ECHO</Attribution>
      <SyndicationRight>open</SyndicationRight>
</OpenSearchDescription>
```

OSDD without parameter extension

```
<?xml version="1.0"?>
<OpenSearchDescription
      xmlns="http://a9.com/-/spec/opensearch/1.1/"
      xmlns:echo="http://www.echo.nasa.gov/esip"
      xmlns:geo="http://a9.com/-/opensearch/extensions/geo/1.0/"
      xmlns:time="http://a9.com/-/opensearch/extensions/time/1.0/"
      xmlns:esipdiscovery="http://commons.esipfed.org/ns/discovery/1.2/
" esipdiscovery:version="1.2"
      xmlns:params="http://a9.com/-
/spec/opensearch/extensions/parameters/1.0/"
      xmlns:atom="http://www.w3.org/2005/Atom" >
      <ShortName>ECHO Dataset Search</ShortName>
      <Description>NASA ECHO Dataset search using geo, time and
parameter extensions</Description>
      <Tags>ECHO NASA CWIC CEOS ESIP OGC dataset</Tags>
      <Contact>support@echo.nasa.gov</Contact>
      <Url
            type="application/atom+xml"
            params:method="GET"
            template="https://api.echo.nasa.gov:443/opensearch/datasets
            .atom?keyword={searchTerms?}&amp;instrument={echo:instrumen
            t?}&amp;satellite={echo:satellite?}&amp;boundingBox={geo:bo
            x?}&amp;geometry={geo:geometry?}&amp;placeName={geo:name?}&
            amp;startTime={time:start?}&amp;endTime={time:end?}&amp;cur
            sor={startPage?}&amp;numberOfResults={count?}&amp;uid={geo:
            uid?}&amp;clientId=foo">
      </Url>
      <Query
            role="example"
            echo:instrument="AMSR-E"
            echo:satellite="Aqua"
            title="Sample search"
            geo:box="-180.0,-90.0,180.0,90.0"
            time:start="2002-05-04T00:00:00-0400"
            time:stop="2009-05-04T00:00:00-0400"/>
      <Attribution>NASA ECHO</Attribution>
      <SyndicationRight>open</SyndicationRight>
</OpenSearchDescription>
```

ATOM result

```
<?xml version="1.0" encoding="UTF-8"?>
<feed esipdiscovery:version="1.2"
      xmlns=”http://www.w3.org/2005/Atom”
      xmlns:dc="http://purl.org/dc/terms/"
      xmlns:echo="http://www.echo.nasa.gov/esip"
      xmlns:esipdiscovery="http://commons.esipfed.org/ns/discovery/1.2/
      "
      xmlns:georss="http://www.georss.org/georss/10"
      xmlns:gml="http://www.opengis.net/gml" xmlns:os="http://a9.com/-
      /spec/opensearch/1.1/" xmlns:time="http://a9.com/-
      /opensearch/extensions/time/1.0/">
      <updated>2014-08-01T17:22:04.288Z</updated>
      <id>https://api.echo.nasa.gov:443/opensearch/datasets.atom</id>
      <author>
            <name>ECHO</name>
            <email>support@echo.nasa.gov</email>
      </author>
      <title type="text">ECHO dataset metadata</title>
      <os:totalResults>3434</os:totalResults>
      <os:itemsPerPage>1</os:itemsPerPage>
      <os:startIndex>1</os:startIndex>
      <Query searchTerms="FIFE" role="request"/>
      <subtitle type="text">Search for ‘FIFE’</subtitle>
      <link
            href="https://api.echo.nasa.gov:443/opensearch/granules/des
            criptor_document.xml"
            hreflang="en-US"
            rel="search"
            type="application/opensearchdescription+xml"/>
      <link
            href="https://api.echo.nasa.gov/opensearch/datasets.atom?
            keyword=FIFE&numberOfResults=1&cursor=1"
            hreflang="en-US"
            rel="self"
            type="application/atom+xml"/>
      <link
            href="https://api.echo.nasa.gov/opensearch/datasets.atom?
            keyword=FIFE&numberOfResults=1&cursor=3434"
            hreflang="en-US"
            rel="last"
            type="application/atom+xml"/>
      <link
            href="https://api.echo.nasa.gov/opensearch/datasets.atom?
            keyword=FIFE&numberOfResults=1&cursor=2"
            hreflang="en-US"
            rel="next"
            type="application/atom+xml"/>
      <link href="https://api.echo.nasa.gov/opensearch/datasets.atom?
            keyword=FIFE&numberOfResults=1&cursor=2"
            hreflang="en-US"
            rel="first"
            type="application/atom+xml"/>
      <link
```

```
        href="https://wiki.earthdata.nasa.gov/display/echo/Open+Sea
        rch+API+release+information"
        hreflang="en-US"
        rel="describedBy"
        title="Release Notes"
        type="text/html"/>
  <entry>
        <id>https://api.echo.nasa.gov:443/opensearch/datasets.atom?
        uid=C179003030-ORNL_DAAC</id>
        <author>
              <name>ECHO</name>
              <email>support@echo.nasa.gov</email>
        </author>
        <title type="text">15 Minute Stream Flow Data: USGS
        (FIFE)</title>
        <summary type="text">ABSTRACT: USGS 15 minute stream flow
        data for Kings Creek on the Konza Prairie</summary>
        <updated>2008-12-02T00:00:00.000Z</updated>
        <echo:datasetId>15 Minute Stream Flow Data: USGS
        (FIFE)</echo:datasetId>
        <echo:shortName>doi:10.3334/ORNLDAAC/1</echo:shortName>
        <echo:versionId>1</echo:versionId>
        <echo:dataCenter>ORNL_DAAC</echo:dataCenter>
        <echo:processingLevelId>3</echo:processingLevelId>
        <link
              href="http://daac.ornl.gov/cgi-
              bin/dsviewer.pl?ds_id=1"
              hreflang="en-US"
              rel="enclosure"/>
        <link
              href="http://daac.ornl.gov/FIFE/guides/15_min_strm_fl
              ow.html"
              hreflang="en-US"
              rel="describedBy"
              title="USGS 15 minute stream flow data for Kings
              Creek on the Konza Prairie (VIEW RELATED
              INFORMATION)"/>
        <georss:point>39.1 -96.6</georss:point>
        <georss:box>39.1 -96.6 39.1 -96.6</georss:box>
        <link
              href="http://gcmd.nasa.gov/getdif.htm?FIFE_STRM_15M"
              hreflang="en-US"
              rel="enclosure"
              title="doi:10.3334/ORNLDAAC/1" type="text/html"/>
        <link
              href="https://api.echo.nasa.gov:443/opensearch/granul
              es.atom?clientId=our_html_ui&amp;shortName=doi:10.333
              4/ORNLDAAC/1&amp;versionId=1&amp;dataCenter=ORNL_DAAC
              "
              hreflang="en-US"
              rel="search"
              title="Search for granules"
              type="application/atom+xml"/>
        <link
              href="https://api.echo.nasa.gov:443/opensearch/granul
              es/descriptor_document.xml?clientId=our_html_ui&amp;s
              hortName=doi:10.3334/ORNLDAAC/1&amp;versionId=1&amp;d
              ataCenter=ORNL_DAAC"
```

```
            hreflang="en-US"
            rel="search"
            title="Custom ECHO Granule OpenSearch Descriptor
            Document"
            type="application/opensearchdescription+xml"/>
      <link
            href="https://api.echo.nasa.gov:443/catalog-
            rest/echo_catalog/datasets/C179003030-ORNL_DAAC.xml"
            hreflang="en-US"
            rel="alternate"
            title="Product metadata"
            type="application/xml"/>
            <dc:date>1984-12-25T00:00:00.000Z/1988-03-
            04T00:00:00.000Z</dc:date>
  </entry>
</feed>
```