CMR Harvesting Best Practices

Prior to harvesting CMR metadata, please send a request to ESDIS management via email to: support@earthdata.nasa.gov

Both NASA EOSDIS data providers and external organizations require mechanisms to enable harvesting the metadata in NASA's Common Metadata Repository. While NASA EOSDIS data providers are often looking for both collection (dataset) and granule metadata in order to validate the CMR's holdings against their local archives, other organizations will primarily focus on harvesting collection level metadata to copy into their own unique repositories.

The CMR's harvesting functionality was implemented to better the use cases in a way that the following criteria are met:

1. Large result sets can be retrieved
2. While iterating through result sets, the results remain consistent
3. Performance of other queries in the system are unaffected
4. Changes to inventory are easily discoverable

In order to meet those criteria, an important new concept named Search After is introduced. After introducing the Search After concept we will walk through 3 categories of use cases related to harvesting:

1. Populating External Systems
2. Capturing Inventory Changes
3. Synchronizing External Systems with the CMR

CMR Search Basics

This document assumes that the reader is familiar with basic usage of the CMR Search API. You can find more information about the CMR Search API here - CMR Client Partner User Guide, and API documentation here - https://cmr.earthdata.nasa.gov/search/site/docs/search/api.html.

Search After

Before jumping into the use cases, we should first introduce the Search After feature on the CMR Search API. Many of the use cases will involve iterating through large results sets of data, and the way that is accomplished via the CMR APIs is by using the search-after parameter. Search-after is the Elasticsearch replacement of scrolling. Here is Elasticsearch's note on scrolling: We no longer recommend using the scroll API for deep pagination. If you need to preserve the index state while paging through more than 10,000 hits, use the `search_after` parameter with a point in time (PIT). The reason Elasticsearch deprecated scroll is that scroll is too resource intensive and it is easy for one imprudent client to affect the whole ES cluster health via scrolling requests.

Besides scrolling, those familiar with the CMR API have potentially used the paging parameters **page_num** and **page_size** or **offset**. There are two downsides to paging that make it not suited well for harvesting:

1. In between search calls data can be ingested or deleted which means that the same result may show up in multiple calls or data can be missed when going through the pages.
2. Deep paging is inefficient for our Elasticsearch system. Queries for a high offset will take longer and can impact the performance of other queries in the system. As a result the CMR rejects requests for paging beyond the one millionth result for a search.

Search-after addresses both of these issues and is the recommended way to harvest CMR metadata. Search-after is supported via `CMR-Search-After` header and it can be used on both the collections and granules search endpoints.

Search After is only supported for parameter queries and JSON queries. All query parameters are available with the exception of the `page_num` and `offset` parameters.

Search After is stateless, it is always resolved against the latest version of the data. Any search against CMR that has results not fully returned in the current request will return a `search-after` value in the `CMR-Search-After` header of the search response. User can then pass this returned value in the `CMR-Search-After` header of the following request to retrieve the next page of result based on the specified page_size. Each search request will result in a new `search-after` value returned in the `CMR-Search-After` response header. Supplying the new `search-after` value in the following request's `CMR-Search-After` header will retrieve the next page. The `CMR-Hits` header is useful for determining the number of requests that will be needed to retrieve all the available results.

When all the results have been returned, the subsequent search will return an empty result set and no `CMR-Search-After` header in the response.

Different from scrolling requests, each search-after request needs to supply all the search parameters, and the `CMR-Search-After` header needs to be updated with the new value returned from the previous search to page through the whole result set. Although user can change the search parameters and still get results back as long as the sort order of the search is unchanged, it breaks the rationale of paging and offers no real use case. Thus user should always supply the same search parameters while using Search After requests to page through a large result set.

Example

As a harvesting client, I want to retrieve all collection metadata.

```
curl -i -H "Client-Id: <client id>" "https://cmr.earthdata.nasa.gov/search
/collections?page_size=2000"
```

We are using a page size of 2000 because that is the maximum allowed page size for a single request for the CMR. This query will return the first 2000 results. There are two headers returned that I will pay attention to:

CMR-Hits: 32636 tells me how results there are total. I know that I will need to scroll 16 times to get through all of the results. I don't actually need to know this because I will just scroll until no more results are returned, but can be useful for someone just wanting to know how many results there are.

CMR-Search-After: ["anomalies of monthly total precipitation (percentage of norm)","SCIOPS","OSADKUZL","not provided",1214608709,5] gives me back the search after value that I will need to supply to my next subsequent query.

Here is the query I will use to retrieve the next result set:

```
curl -i -H "Client-Id: <client id>" -H 'CMR-Search-After: ["anomalies of
monthly total precipitation (percentage of norm)","SCIOPS","OSADKUZL","not
provided",1214608709,5]' "https://cmr.earthdata.nasa.gov/search
/collections?page_size=2000"
```

This query will return the next 2000 collections and a new **CMR-Search-After** header value which I can pass onto the next request to retrieve the next 2000 matching collections.

My queries are all returning collection references. If I wanted to choose a different format I would use a different extension:

For example for the original metadata format for every collection I can call https://cmr.earthdata.nasa.gov/search/collections.**native** or for echo10 https://cmr.earthdata.nasa.gov/search/collections.**echo10**.

The full list of supported formats is documented here: https://cmr.earthdata.nasa.gov/search/site/docs/search/api.html#supported-result-formats

Populating External Systems

This category of use cases is for clients want to retrieve all or some subset of the metadata from the CMR in order to support an initial load of their system.

Collection Use Cases

As a harvesting client, I want to retrieve all collection metadata.

Documented in scrolling example above.

As a harvesting client, I want to retrieve all collection metadata based on a given tag (CWIC, FedEO)

Refer to the CMR Search API Documentation for a detailed explanation of tag features.

1. Initiate request

   ```
   curl -i -H "Client-Id: <client id>" "https://cmr.earthdata.nasa.gov
   /search/collections?tag_key=org.ceos.wgiss.cwic.
   quality&page_size=2000"
   ```

2. Search After until no more results are returned

```
curl -i -H "Client-Id: <client id>" -H 'CMR-Search-After: <search
after value>' https://cmr.earthdata.nasa.gov/search/collections?
tag_key=org.ceos.wgiss.cwic.quality&page_size=2000
```

where <search after value> is returned in the header of the previous search after request.

Granule Use Cases

As a harvesting client, I want to retrieve all granule metadata for a given collection

Using C193529899-LPDAAC_ECS as the example collection.

1. Initiate request

```
curl -i -H "Client-Id: <client id>" "https://cmr.earthdata.nasa.gov
/search/granules?collection_concept_id=C193529899-
LPDAAC_ECS&page_size=2000"
```

2. Search After until no more results are returned

```
curl -i -H "Client-Id: <client id>" -H 'CMR-Search-After: <search
after value>' "https://cmr.earthdata.nasa.gov/search/granules?
collection_concept_id=C193529899-LPDAAC_ECS&page_size=2000"
```

where <search after value> is returned in the header of the previous search after request.

Capturing Inventory Changes

Clients want to make sure they have the most recent data and have removed any data that is no longer valid.

Note that depending on the harvesting interval many of the collection searches will most likely return less than 2000 hits and as such Search After is not required the majority of times. In this case, the **CMR-Hits** h eader would return a value that is less than the provided page_size. You can stop the search there or you can still submit a follow-up Search After request, and CMR would simply return an empty result set for the follow-up Search After request.

Collection Use Cases

As a harvesting client, I want to retrieve only the collection metadata which was revised after a given date.

1. Initiate request

```
curl -i -H "Client-Id: <client id>" "https://cmr.earthdata.nasa.gov
/search/collections?updated_since=2017-07-12T20:06:38.331
Z&page_size=2000"
```

2. Search After until no more results are returned

```
curl -i -H "Client-Id: <client id>" -H 'CMR-Search-After: <search
after value>' "https://cmr.earthdata.nasa.gov/search/collections?
updated_since=2017-07-12T20:06:38.331Z&page_size=2000"
```

where <search after value> is returned in the header of the previous request.

As *a harvesting client, I want to retrieve only the collection metadata which was newly added to the CMR after a given date.*

1. Initiate request

```
curl -i -H "Client-Id: <client id>" "https://cmr.earthdata.nasa.gov
/search/collections?created_at=2017-07-12T20:06:38.331
Z&page_size=2000"
```

2. Search After until no more results are returned

```
curl -i -H "Client-Id: <client id>" -H 'CMR-Search-After: <search
after value>' "https://cmr.earthdata.nasa.gov/search/collections?
created_at=2017-07-12T20:06:38.331Z&page_size=2000"
```

   where <search after value> is returned in the header of the previous request.

As a harvesting client, I want to identify the collection metadata which was deleted after a given date.

See https://cmr.sit.earthdata.nasa.gov/search/site/docs/search/api.html#deleted-collections for more details.

```
curl -i -H "Client-Id: <client id>" "https://cmr.earthdata.nasa.gov/search
/deleted-collections?revision_date=2017-07-12T20:06:38.331Z
```

Granule Use Cases

As a harvesting client, I want to tell which collections have added granules since the last time I harvested.

1. Initiate request

```
curl -i -H "Client-Id: <client id>" "https://cmr.earthdata.nasa.gov
/search/collections?has_granules_created_at=2017-07-12T20:06:38.331Z,
&page_size=2000"
```

2. Search After until no more results are returned

```
curl -i -H "Client-Id: <client id>" -H 'CMR-Search-After: <search
after value>' "https://cmr.earthdata.nasa.gov/search/collections?
has_granules_created_at=2017-07-12T20:06:38.331Z,&page_size=2000"
```

   where <search after value> is returned in the header of the previous request.

As a harvesting client, I want to tell which collections have added or updated granules since the last time I harvested.

1. Initiate request

```
curl -i -H "Client-Id: <client id>" "https://cmr.earthdata.nasa.gov
/search/collections?has_granules_revised_at=2017-07-12T20:06:38.331Z,
&page_size=2000"
```

2. Search After until no more results are returned

```
curl -i -H "Client-Id: <client id>" -H 'CMR-Search-After: <search
after value>' "https://cmr.earthdata.nasa.gov/search/collections?
has_granules_revised_at=2017-07-12T20:06:38.331Z,&page_size=2000"
```

where <search after value> is returned in the header of the previous request.

As a harvesting client, I want to retrieve only the granule metadata which was newly added to the CMR after a given date.

1. Find collections which have newly added granules (same as the use case above "As a harvesting client, I want to tell which collections have added granules since the last time I harvested").
    a. Initiate request to capture collection-ids

    ```
    curl -i -H "Client-Id: <client id>" "https://cmr.earthdata.
    nasa.gov/search/collections?has_granules_created_at=2017-07-
    12T20:06:38.331Z,&page_size=2000"
    ```

    b. Scroll until no more results are returned

    ```
    curl -i -H "Client-Id: <client id>" -H 'CMR-Search-After:
    <search after value>' "https://cmr.earthdata.nasa.gov/search
    /collections?has_granules_created_at=2017-07-12T20:06:38.331Z,
    &page_size=2000"
    ```

    where <search after value> is returned in the header of the previous request.

2. Initiate request to retrieve granule results (pass in the collection IDs captured from step 1).

```
curl -i -g -H "Client-Id: <client id>" "https://cmr.earthdata.nasa.
gov/search/granules?created_at=2017-07-12T20:06:38.331
Z&collection_concept_id[]=C1214471197-ASF&collection_concept_id[]
=C1214470533-ASF&page_size=2000"
```

3. Search After until no more results are returned

```
curl -i -H "Client-Id: <client id>" -H 'CMR-Search-After: <search
after value>' "https://cmr.earthdata.nasa.gov/search/granules?
created_at=2017-07-12T20:06:38.331Z&collection_concept_id[]
=C1214471197-ASF&collection_concept_id[]=C1214470533-
ASF&page_size=2000"
```

where <search after value> is returned in the header of the previous request.

As a harvesting client, I want to retrieve only the granule metadata which was revised after a given date.

Currently there is not a way to check for collections which have had granule metadata created or revised after a given date (only newly created). The client will need to provide either a list of collection concept-ids or a provider id to limit the search against.

Example with provider-id of ASF:

1. Initiate request

```
curl -i -H "Client-Id: <client id>" "https://cmr.earthdata.nasa.gov
/search/granules?updated_since=2017-07-12T20:06:38.331
Z&provider=ASF&page_size=2000"
```

2. Search After until no more results are returned

```
curl -i -H "Client-Id: <client id>" -H 'CMR-Search-After: <search
after value>' "https://cmr.earthdata.nasa.gov/search/granules?
updated_since=2017-07-12T20:06:38.331Z&provider=ASF&page_size=2000"
```

   where <search after value> is returned in the header of the previous request.

3. Repeat the steps for each provider the client is harvesting.

Example with a list of collection concept IDs:

1. Initiate request passing in all of the collection concept IDs the client is interested in. Note that the CMR will also accept a POST with a Content-type of application/x-www-form-urlencoded if there are a large number of collection concept IDs to pass in.

```
curl -i -H "Client-Id: <client id>" "https://cmr.earthdata.nasa.gov
/search/granules?updated_since=2017-07-12T20:06:38.331
Z&collection_concept_id[]=C1214471197-ASF&collection_concept_id[]
=C1214470533-ASF&page_size=2000"
```

2. Scroll until no more results are returned

```
curl -i -H "Client-Id: <client id>" -H 'CMR-Search-After: <search
after value>' "https://cmr.earthdata.nasa.gov/search/granules?
updated_since=2017-07-12T20:06:38.331Z&collection_concept_id[]
=C1214471197-ASF&collection_concept_id[]=C1214470533-
ASF&page_size=2000"
```

   where <search after value> is returned in the header of the previous request.

As a harvesting client, I want an Atom feed of new and updated granules.

Identical to the use case above "As a harvesting client, I want to retrieve only the granule metadata which was revised after a given date" except the calls to the granules endpoints should use the extension **.atom**.

For example:

```
curl -i -H "Client-Id: <client id>" -H 'CMR-Search-After: <search after
value>' "https://cmr.earthdata.nasa.gov/search/granules.atom?
updated_since=2017-07-12T20:06:38.331Z&collection_concept_id[]=C1214471197-
ASF&collection_concept_id[]=C1214470533-ASF&page_size=2000"
```

As a harvesting client, I want to identify the granule metadata which was deleted after a given date.

```
curl -i -H "Client-Id: <client id>" -H 'CMR-Search-After: <search after
value>' "https://cmr.earthdata.nasa.gov/search/deleted-granules.json?
revision_date=2017-01-20T00:00:00Z&collection_concept_id[]=C1214471197-
ASF&page_size=2000"
```

Synchronizing External Systems with the CMR

Data providers have use cases which involve ensuring an external system is synchronized with the CMR. Use cases were captured from providers comments on Granule Reconciliation with CMR. All of the examples will use ASF as the provider.

Collection Use Cases

As a Data Provider, I want to ensure all of my collections are available in the CMR

1. Capture all of the entry titles for collections in your system.
2. Capture all of the entry titles for collections in the CMR. Note that the entry-title can be captured from multiple formats. In the XML references response the entry title is in the <**name>** tag. In the JSON response the entry title is in the **title** field.
    a. Initiate request

    ```
    curl -i -H "Client-Id: <client id>" "https://cmr.earthdata.
    nasa.gov/search/collections?provider=ASF&page_size=2000"
    ```

    b. Search After until no more results are returned

    ```
    curl -i -H "Client-Id: <client id>" -H 'CMR-Search-After:
    <search after value>' "https://cmr.earthdata.nasa.gov/search
    /collections?provider=ASF&page_size=2000"
    ```

    where <search after value> is returned in the header of the previous request.

3. Diff the two lists of entry titles.
4. Ingest any collections which were missing from the CMR.

As a Data Provider, I want to ensure there are no collections available in the CMR which are no longer in my system

The process is identical to the above use case except in step 4 the provider should delete the extra collections from the CMR.

As a Data Provider, I want to ensure all of my collections are up to date in the CMR

1. Capture the entry title and last update time for all collections in your system.
2. Capture the entry title and last update time for all collections in the CMR using the JSON format. The entry title is in the **title** field and last update is the **updated** field.
    a. Initiate request

    ```
    curl -i -H "Client-Id: <client id>" "https://cmr.earthdata.
    nasa.gov/search/collections.json?provider=ASF&page_size=2000"
    ```

    b. Search After until no more results are returned

    ```
    curl -i -H "Client-Id: <client id>" -H 'CMR-Search-After:
    <search after value>' "https://cmr.earthdata.nasa.gov/search
    /collections.json?provider=ASF&page_size=2000"
    ```

    where <search after value> is returned in the header of the previous request.

3. Compare the entry title and last update time for every collection in the two systems.
4. Ingest any collections for entry titles that were missing from the CMR or have a last update time not equal to the last update time in the source system.

Granule Use Cases

As a Data Provider, I want to ensure all of my granules are available in the CMR

For the best performance check all of the granules one collection at a time rather than for all granules for your provider at once.

1. Capture all of the granule URs for granules for the given collection in your system.
2. Capture all of the granule URs for granules for the given collection in the CMR. Note that the granule UR can be captured from multiple formats. In the XML references response the granule UR is in the <**name>** tag. In the JSON response the granule UR is in the **title** field.
    a. Initiate request

```
curl -i -H "Client-Id: <client id>" "https://cmr.earthdata.
nasa.gov/search/granules?collection_concept_id[]=C1214471197-
ASF&page_size=2000"
```

b. Search After until no more results are returned

```
curl -i -H "Client-Id: <client id>" -H 'CMR-Search-After:
<search after value>' "https://cmr.earthdata.nasa.gov/search
/granules?collection_concept_id[]=C1214471197-
ASF&page_size=2000"
```

where <search after value> is returned in the header of the previous request.

3. Diff the two lists of granule URs.
4. Ingest any granules which were missing from the CMR.
5. Repeat the same process for each collection.

As a Data Provider, I want to ensure there are no granules available in the CMR which are no longer in my system

The process is identical to the use case above except in step 4 the provider should delete the extra granules from the CMR.

As a Data Provider, I want to ensure all of my granules are up to date in the CMR

For the best performance check all of the granules one collection at a time rather than for all granules for your provider at once.

1. Capture the granule UR and last update time for all granules for the given collection in your system.
2. Capture the granule UR and last update time for all granules for the given collection in the CMR using the JSON format. The granule UR is in the **title** field and last update is the **updated** field.
    a. Initiate request

```
curl -i -H "Client-Id: <client id>" "https://cmr.earthdata.
nasa.gov/search/granules.json?collection_concept_id[]
=C1214471197-ASF&page_size=2000"
```

b. Search After until no more results are returned

```
curl -i -H "Client-Id: <client id>" -H 'CMR-Search-After:
<search after value>' "https://cmr.earthdata.nasa.gov/search
/granules.json?collection_concept_id[]=C1214471197-
ASF&page_size=2000"
```

where <search after value> is returned in the header of the previous request.

3. Compare the granule UR and last update time for every granule for the collection in the two systems.
4. Ingest any granules for granule URs that were missing from the CMR or have a last update time not equal to the last update time in the source system.
5. Repeat the same process for each collection.