# History on Demand (HoD) - Archive - v3

Domain Portfolio: Conditions | Domain: Historical | Usage Classification: **Limited Availability**

**Geography:** Global **Attribution Required:** No **Attribution Requirements:** N/A

## Overview

### The Intent of the History on Demand APIs

The History on Demand (HoD) APIs provide access to global, historical weather data. Historical weather data is prevalent in trend analysis and the training of analytical models pertaining to energy, agriculture, insurance and many other industries. These APIs serve the same data, but allow you to collect that data the way that works best for you. Each has its own style of interaction, allowable query types, delivery mode, and restrictions. Select the API that suits your usage style, desired data format and delivery mechanism. **HoD Archive** (this API) is an asynchrous approach, and **HoD Direct** (the other API) is synchronous. Information common to both is centralized within one document, which is linked, where necessary, from within each respective API document.

### The API

**HoD Archive** is an asynchronous API. Requesting data from this API can be thought of as *creating* a data retrieval job. This is followed by checking on the job's progress, and ultimately retrieving your data from an IBM Cloud Object Storage location you specify. Nearest grid-neighbor geopoint, multi-point, and bounding box query types are available. See the Data Layers section in the HoD Common Information document for details on data layers served by the HoD APIs.

### Setup

Using the HoD Archive API's asynchronous interaction model will require some setup for clients. In a synchronous interaction, the data is returned nearly immediately in the response to the GET request that asked for it. In the asynchronous interaction, the request for the data is specified in the first call, and at some later point, the data is prepared and delivered to a predetermined location. That location will be an IBM Cloud Object Storage (COS) bucket the caller owns and maintains. This means that the first setup step is to acquire an IBM Cloud account and set up the COS bucket to be used as a receptacle for data retrieval results. The details of this setup process can be found below in the IBM Cloud Account and Cloud Object Storage setup section. Once the setup is done, and you've acquired an apiKey with which to call the HoD Archive API, the process is relatively simple. You specify the details of the data retrieval you'd like to perform in the body of a POST request, check its status, and examine your results once they've arrived in your bucket.

### If you are experiencing difficulty

First, check the Known Issues section of this document, to assess whether what you are experiencing is accounted for there. If not, please send a description of the behavior to support.

### Additional Considerations Regarding the Delivered Data

Because your data will be delivered to an IBM Cloud Object Storage bucket that you own, you have the liberty of performing subsequent analysis on that data either with your own tools, or within the IBM Cloud environment. Some data scientists and analysts prefer to download the data to their local platforms and use their own tools. This is simply a reminder that there is an alternative approach. One could upload relevant data to another COS bucket and use IBM Cloud tools like SQL Query Service to perform further analysis extracting relationships between that data and the weather data.

## Caching and Content Type

Caching is not used in any way with this API.  All API call responses are JSON.  The format of the data delivered is dictated by the specification of the retrieval request.

## Special Considerations Regarding Your Use of this API

- It is **very important** to understand how this API differs from its synchronous ancestor, and what those differences imply regarding the best way to approach using it.  For more, see the Best Practices section.
- We have prepared a Postman Collection for your convenience in interacting with the API.  Feel free to download it, and customize it for your needs.  The collection contains a sample query to test your configuration.  It will verify your COS bucket is visible and writable by the API (has been assigned the correct permissions), your API key has access to the API, and you are able to execute data retrievals and see results arrive in your COS bucket
- If this is your first time accessing the API, we strongly recommend you review the Example API Interaction for a walkthrough in how to use HoD Archive.
- The Data Layers section in the HoD Common Information document describes the data layers currently available in the HoD Archive API.
- The IBM Cloud Account and Cloud Object Storage Setup section describes getting set up to receive data from your queries.
- For a complete understanding of the rules and interpretation on date and date/time expressions sent to the API, see the Date / Time Inputs and Interpretation section in the HoD Common Information document.
- For information on specifying a location (using Well-Known Text) for your data retrieval, see the Location Specification section in the HoD Common Information document.
- For data storage cost concerns, see Data Storage Costs.
- Calls to this API will record usage for your API key, viewable through the /usage endpoint. See the HoD Usage section in the HoD Common Information document.
- If you plan to make very large queries, Parquet is the preferred file type, as it takes up less storage space and takes less time to write than the other file types that are supported.
- The HoD Archive API will apply backpressure.  For more information, see Backpressure
- A Frequently Asked Questions document is available for additional questions

## Restrictions

- Bounding box queries will be limited to a maximum 45° longitude by 45° latitude
- Result sets will be split up into files of 1 million rows or less.
- The date range for any given query is limited to 5 years, and is limited by the current date and time.  This API does not serve forecast data.
- Multi-point queries are limited to sets of 500 points.
- Each caller (each API Key holder) is limited to a configured number of "active" queries.  For details, see Backpressure
- HoD Archive Jobs in general are under a constraint imposed by an upstream system which limits the max time a job can compute.
    - This is the reason for the time range and spatial limitations imposed.
    - If the time limit is exceeded, the job will appear to the client to “**error**,” with a status of “**Internal Server Error**.”
    - While we seek to mediate this limitation, it is advisable for clients who see an errored job that had a status of **in_progress** for approximately 1 hour, to retry a job reduced in scope either spatially or temporally.

## Known Issues with HoD Archive

- All icon_code data is invalid.  The icon_code_extended data is valid, and we do have the translation table, so if icon_code is important to you, please contact support.  This will be fixed in the future.
- Jobs will infrequently complete, reporting successful execution, but delivering no data to the specified destination bucket. Retry jobs like this where data was expected.  This will be fixed. Contact support with questions.

## Recent Improvements

**2021-11-10**

- Multi-point performance improvement - limitation increased to 500
- Fixed issue where some queries were timing out in under 1 hour
- Fixed rounding issues that made some values, in certain circumstances, differ from HoD Direct query results
- API no longer produces validations of date ranges outside the available data.  Instead it simply constrains the query to the available data.

## Atomic Endpoints

| |
|---|
| v3/wx/hod/r1/archive |
| v3/wx/hod/r1/activity |
| v3/wx/hod/r1/usage |

## URL Construction

**Request Data (technically “create a data retrieval job”):**
**Method: POST**
**Required Request Parameters:** apiKey
**Required Post Body:** location, startDateTime, endDateTime, format, units, resultsLocation
https://api.weather.com/v3/wx/hod/r1/archive?apiKey=**yourApiKey**

https://api.weather.com/v3/wx/hod/r1/archive?apiKey=**yourApiKey** (with POST body as shown below)

**POST Body Example:**

```
{
  "location": "POINT (44.365 -90.445)",
  "startDateTime": "2016-04-16T00",
  "endDateTime":   "2016-04-17T00",
  "format": "csv",
  "units": "s",
  "resultsLocation": "cos://us-geo/my-bucket/"
}
```

**Request Multiple Activity Records:**
**Method: GET**
**Required Request Parameters:** apiKey
**Optional Request Parameters:** startDateTime, endDateTime, pageNumber, pageSize, sort
https://api.weather.com/v3/wx/hod/r1/activity?startDateTime=<**startDateTime**>&endDateTime=<**endDateTime**>&pageNumber=<**pageNumber**>&pageSize=<**pageSize**>&sort=<**sort**>&apiKey=**yourApiKey**

https://api.weather.com/v3/wx/hod/r1/activity?startDateTime=2021-01-01T00&endDateTime=2021-03-01T00&pageNumber=0&pageSize=25&sort=submissionTime.desc&apiKey=**yourApiKey**

**Request Single Activity Record by Job Id:**
**Method: GET**
**Required Request Parameters:** jobId, apiKey
https://api.weather.com/v3/wx/hod/r1/activity?jobId=<**jobId**>&apiKey=**yourApiKey**

https://api.weather.com/v3/wx/hod/r1/activity?jobId=d0e0463b-2715-4e41-b03b-f3[…]=&apiKey=**yourApiKey**

For Usage URL construction, see the HoD Usage section in the HoD Usage section in the HoD Common Information document.

## Valid Parameter Definitions - Creating a Job

### POST /archive JSON POST body elements

| Tag | Description | Example JSON tag: value |
|---|---|---|
| location | Valid Well-Known-Text spatial geometry representation<br>See Location Specification section in the HoD Common Information document<br>Supports POINT, MULTIPOINT, and BBOX | **“location”: "POINT (-31.427191 -63.982213)"** |
| startDateTime | The beginning of the analysis period. Must be before endDateTime. **Inclusive.** Several Formats accepted, see Date / Time Inputs and Interpretation | **"startDateTime": "2016-01-01"** |
| endDateTime | The end of the analysis period.. Must be after startDateTime. **Exclusive.** Several Formats accepted, see Date / Time Inputs and Interpretation | **"endDateTime": "2016-04-01"** |
| format | The desired format of the resulting data | |
| | Comma-Delimited Values | **"format": "csv"** |
| | JSON Lines | **"format": "jsonl"** |
| | Apache Parquet | **"format": "parquet"** |
| | Optimized Row-Columnar Format | **"format": "orc"** |
| | Avro (Schema-based, compressed, binary format) | **"format": "avro"** |
| units | 1 character representing the unit system in which the response should be expressed:<br>e (english/imperial), m (metric), s (SI). | **“units”: “s”** |
| resultsLocation | valid IBM COS bucket path | **"resultsLocation": "cos://us-geo/my-bucket/"** |

## Valid Parameter Definitions - Activity

### GET /activity?jobId=<jobId>

| Parameter | Description | Req | Example |
|---|---|---|---|
| jobId | ID of the job for which activity data is desired.<br>**`jobId`** parameter is stated as not required because if omitted you are essentially making the **`GET/activity`** call below | N | **jobId=d0e0463b-2715-4e41-b03b-f3[...] (UUID)** |

### GET /activity (multiple)

**Note:** No parameter is required for activity list, since omission results in default values, and all activity instances

| Parameter | Description | Req | Example |
|---|---|---|---|
| startDateTime | The inclusive beginning of the `submissionTime` range from which you would like to see activity history | N | **startDateTime=2021-01-01T00** |
| endDateTime | The exclusive end of the `submissionTime` range from which you would like to see activity history | N | **endDateTime=2021-03-01T00** |
| pageNumber | The desired page number with the first page being number 0 (default 0) | N | **pageNumber=0** |
| pageSize | The maximum number of elements per page (default 20) | N | **pageSize=25** |
| sort | To sort results, add a sort query parameter with the name of the property followed optionally by a comma (,) plus either asc or desc. By default, results will be sorted in descending (desc) order. To sort the results by more than one property, add an additional sort={property} for each additional property. | N | **sort=submissionTime,desc** |

For Usage Valid Parameter Definitions, see the HoD Usage section in the HoD Usage section in the HoD Common Information document.

## Data Elements & Definitions

See the Data Elements & Definitions section in the HoD Common Information document for details on the **Gridded Currents on Demand** response.  However since the **Activity** endpoint is specific to **HoD Archive**, it is included here.

### Multiple Job Activity Call

| Field Name | Description | Type | Range | Sample | Nulls Allowed |
|---|---|---|---|---|---|
| content | array of query metadata | [array] | | | |
| jobId | The ID of the job just submitted via the call | [uuid] | n/a | 123e4567-e89b-12d3-a456-426614174000 | N |
| type | Job type (archive only for now, more in the future) | [string] | archive | archive | N |
| jobStatus | Job Status | [integer] | received, in_progress, complete, error | received | N |
| location | The WKT submitted as the location in the original query | [string] | n/a | "POINT (1.0 2.0)" | Y |
| startDateTime | The inclusive start hour of the data block desired | [ISO] | n/a | "2016-04-16T18:00:00+0000" | Y |
| endDateTime | The exclusive end hour of the data block desired | [ISO] | n/a | "2016-04-17T18:00:00+0000" | Y |
| format | The output format | [string] | csv, jsonl, parquet, orc, avro | csv | Y |
| units | Unit system in which output was written | [string] | m: Metric, e: English, s: SI | m | Y |
| resultsLocation | Location to which the asynchronous result was written | [string] | n/a | "cos://s3.us.cloud-object-storage.appdomain.cloud/my-bucket/weather/2016-04-16/json/jobId=xxxxx" | Y |
| submissionTime | Time query was submitted | [ISO] | n/a | "2020-11-21T15:19:54+0000" | Y |
| completionTime | Time query work was completed / result was delivered | [ISO] | n/a | "2020-11-21T15:20:24+0000" | Y |
| rowsReturned | Number of data rows returned | [integer] | n/a | 13140 | Y |
| usage | Usage units calculated (days touched x points) | [integer] | n/a | 548 | Y |
| pageable | object detailing paging information | [object] | | | |
| totalPages | Total number of pages available | [integer] | n/a | 18 | N |
| totalElements | Total number of elements in all pages | [integer] | n/a | 35 | N |
| pageNumber | Number of current page | [integer] | n/a | 9 | N |
| pageSize | Per-page element limit | [integer] | n/a | 2 | N |
| pageElements | Number of elements on current page | [integer] | n/a | 2 | N |
| first | Is this the first page | [boolean] | true, false | false | N |
| last | Is this the last page | [boolean] | true, false | false | N |
| empty | Is this page empty | [boolean] | true, false | false | N |

### Single Job Activity Call

| Field Name | Description | Type | Range | Sample | Nulls Allowed |
|---|---|---|---|---|---|
| (object) | Single object identical to one "content" object entry from above table for Multiple job Activity call | [object] | | | |

For Usage Data Elements & Definitions, see the HoD Usage section in the HoD Usage section in the HoD Common Information document.

## Example API Interaction

In this section are some simple examples of the ways you can interact with the API.  Note that the **`apiKey`** parameter would be required in all cases.

### Send your data retrieval request.

This is where you will specify the details of the spatial and temporal bounds of the **`/archive`** data you wish to retrieve.
The "**`activeJobs`**" section in the response is covered in Backpressure.

| **REQUEST**<br><br>**`POST /archive`** | **RESPONSE**<br>Requirements for successful response:<br>bucket specification is valid, API has write permissions to it, all of the other values in the retrieval specification are valid values |
|---|---|
| Body example:<br>`{`<br>`  "location": "POINT (44.365 -90.445)",`<br>`  "startDateTime": "2016-04-16T00",`<br>`  "endDateTime":   "2016-04-17T00",`<br>`  "format": "csv",`<br>`  "units": "s",`<br>`  "resultsLocation": "cos://us-geo/my-bucket/"`<br>`}` | `{`<br>`  "job": {`<br>`    "jobId": "(UUID we assign your job)",`<br>`    "type": "archive",`<br>`    "jobStatus": "received",`<br>`    "submissionTime": "2020-11-20T15:21:23+0000"`<br>`  },`<br>`  "activeJobs": {`<br>`    "current": 3,`<br>`    "max": (currently configured max at the time)`<br>`  }`<br>`}` |

## Check on your retrieval request.

Get the details (including the status) by checking the **`/activity`** on the **`jobId`** you saw in the first response.

| **REQUEST**<br>**`GET /activity?jobId=[UUID we assigned your job]`** | **RESPONSE**<br>(sample) |
|---|---|
| | `{`<br>`  "jobId": "(UUID we assign your job)",`<br>`  "type": "archive",`<br>`  "jobStatus": "complete",`<br>`  "location": "POINT (44.365 -90.445)",`<br>`  "startDateTime": "2016-04-16T00:00:00+0000",`<br>`  "endDateTime": "2016-04-17T00:00:00+0000",`<br>`  "format": "csv",`<br>`  "units": "s",`<br>`  "resultsLocation": "cos://s3.us.cloud-object-storage.appdomain.cloud/my-bucket/jobId=(UUID we assign your job)",`<br>`  "submissionTime": "2020-11-20T15:21:23+0000",`<br>`  "completionTime": "2020-11-20T15:21:53+0000",`<br>`  "rowsReturned": 13140,`<br>`  "usage": 548`<br>`}` |

## Retrieve your data

Go to the IBM Cloud Object Storage bucket you specified in the initial request (also reflected in the **`/activity`** response), and you will find the data you requested, in the format you specified, in the folder referenced in the activity response above.  The Cloud Object Storage documentation will be helpful in understanding how you can organize, move or download your data from there.

### Check the details on your activity

Making a call to that same **`/activity`** endpoint *without* specifying a **jobId** will result in a listing of your interactions with the API.  This list can be customized in several ways:

<table>
<tr>
<th><b>REQUEST</b><br><br><b><code>GET /activity ?startDateTime=2021-01-01T00&endDateTime=2021-03-01T00&pageNumber=0&pageSize=20&sort=submissionTime,desc</code></b></th>
<th><b>RESPONSE</b><br><br>(sample)<br>Response contains a list of <b><code>activity</code></b> objects, followed by a <b><code>pageable</code></b> object containing any paging information you may need to inform subsequent requests.  If - in your first request - you sent no <b><code>pageNumber</code></b>, <b><code>pageSize</code></b>, or <b><code>sort</code></b> parameters, the defaults are <b><code>0</code></b>, <b><code>20</code></b>, and <b><code>submissionTime,desc</code></b>, respectively.</th>
</tr>
<tr>
<td></td>
<td><pre>
{
  "content": [
    {
      << Entry identical to single /activity retrieval above >>
    },
    . . .
  ],
  "pageable": {
    "totalPages": 7,
    "totalElements": 134,
    "pageNumber": 0,
    "pageSize": 20,
    "pageElements": 20,
    "first": true,
    "last": false,
    "empty": false
   }
}
</pre></td>
</tr>
</table>

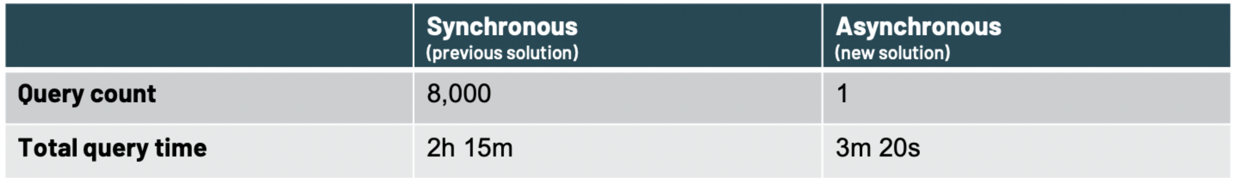
For Usage API Interaction, see the HoD Usage section in the HoD Usage section in the HoD Common Information document.

# Appendices

## Best Practices

**Query efficiency:** Due to the synchronous nature of HoD Conditions, strict record limitations were put in place, forcing many users to make thousands of calls to complete their data request. With an asynchronous interaction and a 40x increase in performance, these limitations are no longer required in HoD Archive. For reference, retrieving one year of data for a bounding box of 40x40 grid points (25,000 $km^2$), takes about 8,000 total queries and over 2 hours to complete via the synchronous HoD Conditions API. That same request in the asynchronous HoD Archive API can be done with a single query, taking just over 3 minutes to complete (see table below).

HoD Conditions vs HoD Archive:
Query count and total query time for a 40x40-gridpoint bounding box and one year of data



| | **Synchronous**<br>(previous solution) | **Asynchronous**<br>(new solution) |
|---|---|---|
| **Query count** | 8,000 | 1 |
| **Total query time** | 2h 15m | 3m 20s |

Therefore, given the asynchronous nature and greater efficiency of the HoD Archive API, you will want to consolidate your data requests into as few queries as possible.

- Example 1: If you were previously making 100 calls for the same point in order to retrieve several years of data, you will now want to reconfigure your query to make one call for that point, over the entire time period you desire.
- Example 2: If you were making requests for several different points over the same time period, you will now want to consider making one request for all points over a given time period, using either a multi-point or bounding box query.

**Backpressure and Retry-After:** HoD Archive is designed to optimally serve its clients, while fairly and evenly distributing its bandwidth. To ensure this, “backpressure” is applied to the caller to control the production of work, which limits the number of concurrent queries that any one caller can run. Once you have hit the limit, subsequent queries will receive an HTTP 429 Too Many Requests response code, with an advisory Retry-After header indicating the number of seconds you should wait before attempting another call. The best programmatic approach to coding against this API is to simply POST your archive retrieval jobs until you get a **`429`**, then adhere to the advised **`Retry-After`** value, and resume sending. For more details, please see Backpressure.

**File Type:** Parquet is the preferred file type for very large queries, as they take up less storage space and take less time to write than the other file types that are supported. You are certainly welcome to use whichever file type fits your needs best, but this is something you will want to keep in mind when requesting large amounts of data as it will have an impact on your query efficiency.

## IBM Cloud Account and Cloud Object Storage Setup

### Sign up for an IBM Cloud account

Begin by creating an IBM Cloud account.

### Create an IBM Cloud Object Storage service instance

As discussed above, HoD Archive will deliver the results of a data retrieval job to a Cloud Object Storage bucket owned and maintained by you. Using your IBM Cloud account, you'll need to create a Cloud Object Storage service instance.

### Create your HoD Archive results buckets

After adding Cloud Object Storage to your account, you'll need to create one or more buckets which you can then use in a request for HoD Archive to deliver your results to. After creating a bucket, save the name because you'll need it later in this process.

### Invite the HoD Archive user to your account

Next, HoD Archive will need permission to write results to your Cloud Object Storage bucket. To accomplish this, you'll first need to invite the HoD Archive user to your account.

1. In your IBM Cloud account, navigate to **Manage** > **Access (IAM)**, and select **Users** on the left.
2. Click **Invite users**.
3. Specify **hdawaupr@us.ibm.com** in the **Enter email addresses** text field.
4. Click **Invite**.

1. **There will be a delay** between the time you invite the user and the time the invite is accepted.  Until that time the invite will show as "**Pending**."  If this delay seems unreasonably long, please contact support.

### Grant the HoD Archive user Writer access to your results bucket

Finally, with the HoD Archive user added to your account, you'll need to grant it permission to write results to one or more buckets in your Cloud Object Storage service instance.

1. In your IBM Cloud account, navigate to **Manage** > **Access (IAM)**, and select **Users** on the left.
2. Select **hdawaupr@us.ibm.com**
3. Select the **Access Policies** tab to assign a new policy.
4. Click **Assign access** to create the new policy.
5. In the dropdown, below “What type of access do you want to assign?” select **Cloud Object Storage** from the services menu.
6. Below “Which services do you want to assign access to?” select **Service based on attributes**.
7. Check the **Service instance** checkbox and select your Cloud Object Storage service instance in the dropdown on the right.
8. Check the **Resource type** checkbox and enter “bucket” (without the quotes) in the text field on the right.
9. Check the **Resource ID** checkbox and enter your bucket name in the text field on the right.
10. Under the “Service access” section check **Writer**.
11. Click **Add**
12. Click **Assign**

### Send results to your bucket

To have results delivered to your bucket you'll specify the bucket's URL as the **resultsLocation** in the POST body of an archive data retrieval request. To get a bucket's url, navigate to your IBM Cloud Dashboard's resource list. Under **Storage** select your Cloud Object Storage instance. This should bring you to a listing of your buckets. Clicking the **Actions (3 dots) menu** of a bucket, choose **Access with SQL** and copy the **Object SQL URL** which should be something like `cos://{region}/{bucket-name}`.

To view and download results after they have been delivered, click the bucket name from your bucket listing. From there you can use the Prefix filter to find the results location identified in the activity record for a request.

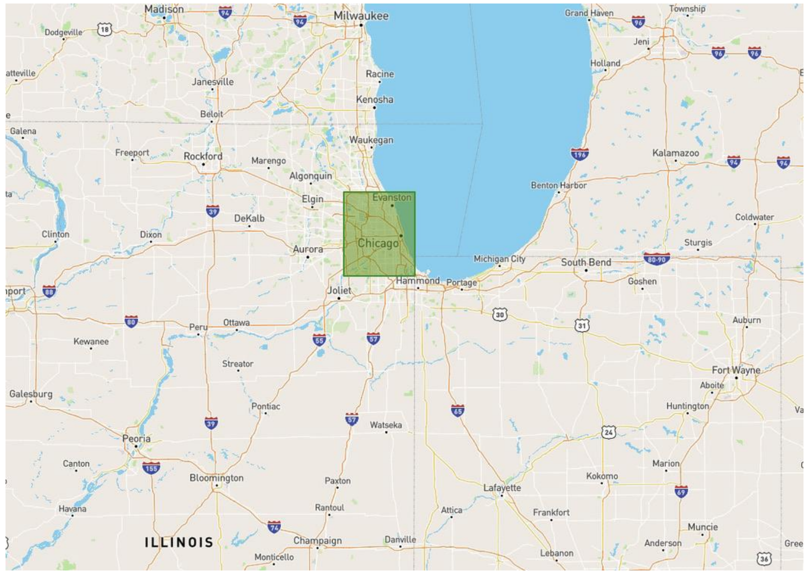
## Data Storage Costs

For approximating the amount of storage required for specific query sizes and file output types, we have provided the following examples.  Please note that these are estimates only and the final results may vary.

| | CSV | JSONL | PARQUET |
|---|---|---|---|
| 1 result row | 257.995 B | 862.094 B | 30.261 B |
| 1 hour | 257.995 B | 862.094 B | 30.261 B |
| 1 year | 2.261 MB | 7.552 MB | 265.1 KB |

### Large City Bounding Box Results Storage

```
BBOX ((41.651367 -88.048553), (42.11758 -87.523956))
```

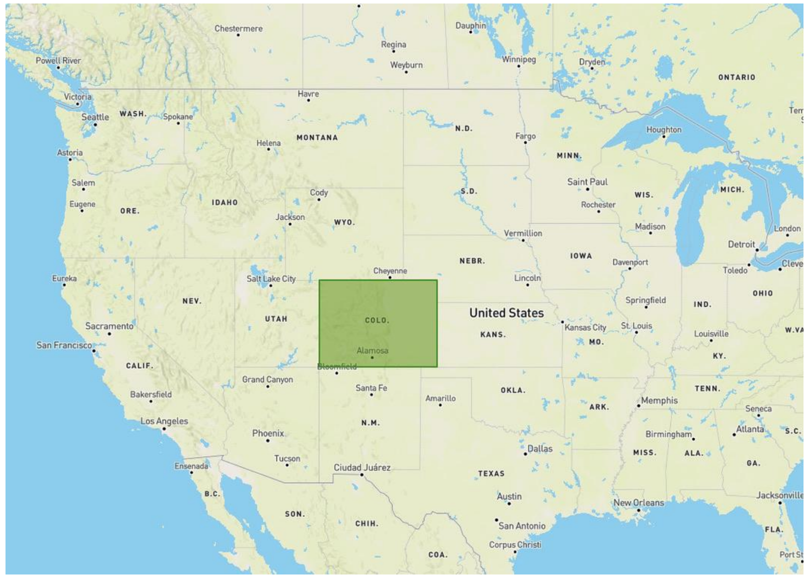


| | CSV | JSONL | PARQUET |
|---|---|---|---|
| 1 result row | 242.890 B | 840.453 B | 19.031 B |
| 1 hour | 29.147 KB | 100.855 KB | 2.284 KB |
| 1 year | 255.326 MB | 883.484 MB | 20.006 MB |

The number of grid points / resultant file size for a Bounding Box request is dependent on the latitude range of the request.

## Colorado Bounding Box Results Storage

```
BBOX ((37.011326 -109.050293), (41.004775 -102.052002))
```



| | CSV | JSONL | PARQUET |
|---|---|---|---|
| 1 result row | 240.607 B | 838.218 B | 16.069 B |
| 1 hour | 3.482 MB | 12.129 MB | 232,502 KB |
| 1 year | 30.497 GB | 106.243 GB | 2.036 GB |

## UK Bounding Box Results Storage

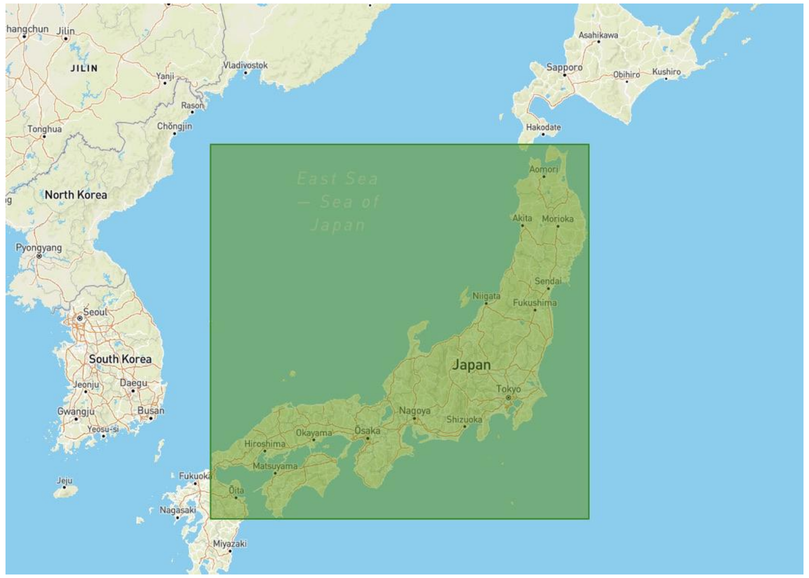
```
BBOX ((49.95122 -8.217773), (61.015725 1.757813))
```

Reykjavík
Tórshavn
Steinkjer
Trondheim
Östersund
Umeå
Ålesund
Sundsvall
Norway
Bergen
Gävle
Oslo
Stavanger
Örebro
Stockholm
Gothenburg
Aalborg
Halmstad
Aberdeen
SCOT.
North Sea
Baltic Sea
Glasgow
Copenhagen
N. IRE.
United Kingdom
Ireland
Kiel
Gdansk
Hamburg
Szczecin
Groningen
Bydgoszcz
Nottingham
WALES
ENG.
Hanover
Berlin
Poland
Netherlands
Cardiff
London
Germany
Dresden
Wroclaw
Belgium
Krakow
Luxembourg
Frankfurt
Czechia
Caen
Paris
Stuttgart
Slovakia
Munich
Vienna
France
Angers
Dijon
Austria
Hungary
Switzerland

| | CSV | JSONL | PARQUET |
|---|---|---|---|
| 1 result row | 238.243 B | 833.585 B | 13.937 B |
| 1 hour | 13.629 MB | 47.685 MB | 797.254 KB |
| 1 year | 119.386 GB | 417.716 GB | 6.984 GB |

## Honshu (Japan) Bounding Box Results Storage

```
BBOX ((32.722599 130.852661), (41.5497 142.075195))
```



| | CSV | JSONL | PARQUET |
|---|---|---|---|
| 1 result row | 238.980 B | 834.400 B | 13.573 B |
| 1 hour | 12.297 MB | 42.935 MB | 698.435 KB |
| 1 year | 107.722 GB | 376.110 GB | 6.119 GB |

## Backpressure

### Definition

HoD Archive is designed to optimally serve all client requests, while fairly and evenly distributing its bandwidth.
In order to do that, the API must ensure that the pace at which it accepts asynchronous requests and submits them for processing is monitored and controlled.
This is referred to as "backpressure" in software design. The optimal way to implement "backpressure" is to control the production of work, or disallow exceeding a certain level of inbound work.

### Implementation

The HoD Archive API will maintain limitations on the number of concurrent queries each caller (each API key holder) is able to run.
That limit will be configurable and is subject to change at any time to accommodate changing conditions within the system and the systems upon which it depends.

### Example

Each time you POST a retrieval job, you will see a section in the response that details your “**`activeJobs`**”.
This section shows your “**`current`**” jobs (in `RECEIVED`, `PENDING`, or `IN_PROGRESS` status),
and the “**`max`**” jobs you are limited to running concurrently.

```
{
  "job": {
    "jobId": "(UUID we assign your job)",
    "type": "archive",
    "jobStatus": "received",
    "submissionTime": "2020-11-20T15:21:23+0000"
  },
  "activeJobs": {
    "current": 3,
    "max": (currently configured max at the time)
  }
}
```

### Triggering Backpressure

Once you have hit the limit (**`current`** = **`max`**), subsequent queries will receive an HTTP 429 Too Many Requests response code, with an advisory Retry-After header indicating the number of **seconds** you should wait before attempting another call. It is important to note, however, that in the background, your “**`current`**” queries are being processed. If upon submitting a query, the “**`activeJobs`**” section shows that you are at capacity, your next query may still be accepted if one of your previous queries completes in the interim.

The “**`activeJobs`**” section in the response is mainly for humans posting individual jobs manually, using curl or Postman. The best programmatic approach to coding against this API is to simply POST your archive retrieval jobs until you get a **`429`**, then adhere to the advised **`Retry-After`** value, and resume sending. There are numerous articles and posts online that illustrate this approach.