hca-cli Documentation

Release 0.1.0

James Mackey, Andrey Kislyuk

May 11, 2020

Contents

This repository is a pip installable Command Line Interface (CLI) and Python library (API) for interacting with the Data Coordination Platform (DCP) of the Human Cell Atlas (HCA).

Currently the *hca* package supports interaction with the Upload Service and Data Storage Service (DSS) for services such as uploading, downloading, and querying data.

The HCA CLI is compatible with Python versions 3.5+ (we are no longer compatible with Python 2.7, and our last compatible Python 2.7 version was *hca==6.4.0*).

CHAPTER 1

Installation

```
pip install hca.
```

CHAPTER 2

Usage

Documentation on readthedocs.io:

- CLI documentation
- Python API documentation

Example CLI/API usage:

- CLI examples (open endpoints)
- CLI examples (restricted endpoints)
- Python API examples (open endpoints)
- Python API examples (restricted endpoints)

To see the list of commands you can use, type `hca --help`.

2.1 Configuration management

The HCA CLI supports ingesting configuration from a configurable array of sources. Each source is a JSON file. Configuration sources that follow the first source update the configuration using recursive dictionary merging. Sources are enumerated in the following order (i.e., in order of increasing priority):

- Site-wide configuration source, `/etc/hca/config.json`
- User configuration source, `~/.config/hca/config.json`
- Any sources listed in the colon-delimited variable `HCA_CONFIG_FILE`
- Command line options

Array merge operators: When loading a chain of configuration sources, the HCA CLI uses recursive dictionary merging to combine the sources. Additionally, when the original config value is a list, the package supports array manipulation operators, which let you extend and modify arrays defined in underlying configurations. See https://github.com/kislyuk/tweak#array-merge-operators for a list of these operators.

2.2 Service to Service Authorization

Google service credentials must be whitelisted before they will authenticate with the HCA CLI.

Set the environment variable *GOOGLE_APPLICATION_CREDENTIALS* to the path of your Google service credentials file to authenticate.

One can also use: `hca dss login`.

See Google service credentials for more information about service accounts. Use the Google Cloud IAM web console to manage service accounts.

CHAPTER 3

Development

To develop on the CLI, first run `pip install -r requirements-dev.txt`. You can install your locally modified copy of the hca package by running `make install` in the repository root directory.

To use the command line interface with a local or test DSS, first run `hca` (or `scripts/hca` if you want to use the package in-place from the repository root directory). This will create the file `~/.config/hca/config.json`, which you can modify to update the value of `DSSClient.swagger_url` to point to the URL of the Swagger definition served by your DSS deployment. Lastly, the CLI enforces HTTPS connection to the DSS API. If you are connecting to a local DSS, make this change in `dcp-cli/hca/util/__init__.py` in the `SwaggerClient` object:

```
scheme = "http"
```

To use the Python interface with a local or test DSS, pass the URL of the Swagger definition to the `DSSClient` constructor via the `swagger_url` parameter:

```
client = DSSClient(swagger_url="https://dss.example.com/v1/swagger.json")
```

You can also layer a minimal config file on top of the default `config.json` using the `HCA_CONFIG_FILE` environment variable, for example:

```
export SWAGGER_URL="https://dss.staging.data.humancellatlas.org/v1/swagger.json"
jq -n .DSSClient.swagger_url=env.SWAGGER_URL > ~/.config/hca/config.staging.json
export HCA_CONFIG_FILE=~/.config/hca/config.staging.json
```

CHAPTER 4

Testing

Before you run tests, first run `hca dss login`. This will open a browser where you can log in to authenticate with Google. Use an email address from one of the whitelisted domains (in `DSS_SUBSCRIPTION_AUTHORIZED_DOMAINS_ARRAY` from here).

Then `make test`.

Primary CI testing is through Travis CI; there is also additional testing with the Gitlab Allspark instance that runs tests for Windows. (Note that Allspark is not open to the public, members of the Human Cell Atlas project can access the Allspark cluster using the Github account associated with the Human Cell Atlas organization on Github.) If submitting PRs that have the potential of being platform-dependent, please ensure the status of "Windows Testing" is verified before merging.

4.1 Bugs

Please report bugs, issues, feature requests, etc. in the HumanCellAtlas/dcp-cli repository on GitHub.

CHAPTER 5

Security Policy

See our Security Policy.

CHAPTER 6

License

Licensed under the terms of the MIT License.

6.1 CLI documentation

Human Cell Atlas Command Line Interface

For general help, run `{prog} help`. For help with individual commands, run `{prog} <command> --help`.

```
usage: hca [-h] [--version] [--log-level {INFO,WARNING,DEBUG,ERROR,CRITICAL}]
           {clear-hca-cache,help,upload,dss,auth} ...
```

6.1.1 Named Arguments

--version	show program's version number and exit
--log-level	Possible choices: INFO, WARNING, DEBUG, ERROR, CRITICAL ['DEBUG', 'INFO', 'WARNING', 'ERROR', 'CRITICAL'] Default: "INFO"

6.1.2 Sub-commands:

6.1.2.1 clear-hca-cache

Clear the cached HCA API definitions. This can help resolve errors communicating with the API.

```
hca clear-hca-cache [-h]
```

6.1.2.2 help

Print help message

```
hca help [-h]
```

6.1.2.3 upload

Upload data to DCP

```
hca upload [-h] {help,select,files,list,areas,forget,creds,status,report} ...
```

6.1.2.3.1 Sub-commands:

help

Display list of upload commands.

```
hca upload help [-h]
```

select

Select upload area to which you wish to upload files.

```
hca upload select [-h] uri_or_alias
```

Positional Arguments

uri_or_alias S3 URI of an upload area, or short alias.

files

Upload a file to the currently selected upload area.

```
hca upload files [-h] [-t <filename>] [--file-extension <fileextension>]
                 [--no-transfer-acceleration] [-q] [-s]
                 <upload_path> [<upload_path> ...]
```

Positional Arguments

<upload_path> Path to files or directories to be uploaded.

Named Arguments

-t, --target-filename Filename to use in upload area (if you wish to change it during upload). Only valid when one file is being uploaded.

--file-extension File extension to limit which files should be uploaded Only valid when directories are targeted for upload.

--no-transfer-acceleration Don't use Amazon S3 Transfer Acceleration. By default we using the aforementioned service to upload via an endpoint geographically close to you, instead of directly to Virginia, USA. However, in some situations this can be slower. Use the S3 Transfer Acceleration Speed Comparison Tool to determine whether you should use this option: https://s3-accelerate-speedtest.s3-accelerate.amazonaws.com/en/accelerate-speed-comparsion.html.

Default: False

-q, --quiet Suppress normal output.

Default: False

-s, --sync If set to true, do not upload files to an area in which the file has already been uploaded before

Default: False

list

List contents of currently selected upload area.

```
hca upload list [-h] [-l]
```

Named Arguments

-l, --long Long listing - show file details.

Default: False

areas

List upload areas the current user has access to. Also see :py:class:`UploadArea`.

```
hca upload areas [-h]
```

forget

Forget about upload area.

```
hca upload forget [-h] uuid_or_alias
```

Positional Arguments

uuid_or_alias Full or partial (alias) UUID of an upload area.

creds

Get/show AWS credentials for access to upload area.

```
hca upload creds [-h] uuid_or_alias
```

Positional Arguments

uuid_or_alias Full or partial (alias) UUID of an upload area.

status

Print status of file in an upload area.

```
hca upload status [-h] [--env ENV] [--uuid UUID] filename
```

Positional Arguments

filename File name

Named Arguments

--env Environment the upload area was created in (default is based on currently selected upload area)

--uuid Full UUID of an upload area (default is based on currently selected upload area)

report

Generate file status report for upload area.

```
hca upload report [-h] [--env ENV] [--uuid UUID]
                  [--output_file_name OUTPUT_FILE_NAME]
```

Named Arguments

--env Environment the upload area was created in (default is based on currently selected upload area)

--uuid Full UUID of an upload area (default is based on currently selected upload area)

--output_file_name Name of output file (default is upload area name)

6.1.2.4 dss

Interact with the HCA Data Storage System

```
hca dss [-h]
        {get-bundles-all,get-bundles-checkout,delete-bundle,get-bundle,patch-bundle,
↪put-bundle,post-bundles-checkout,get-collections,put-collection,delete-collection,
↪get-collection,patch-collection,get-events,get-event,get-file,head-file,put-file,
↪post-search,get-subscriptions,put-subscription,delete-subscription,get-subscription,
↪login,logout,upload,download,download-manifest,create-version,download-collection}
        ...
```

6.1.2.4.1 Sub-commands:

get-bundles-all

Lists all the bundles available in the data-store, responses will be returned in a paginated format, at most 500 values shall be returned at a time. Tombstoned bundles will be omitted from the list of bundles available.

```
hca dss get-bundles-all [-h] --replica {aws,gcp} [--prefix PREFIX]
                        [--token TOKEN] [--per-page PER_PAGE]
                        [--search-after SEARCH_AFTER] [--no-paginate]
```

Named Arguments

--replica Possible choices: aws, gcp

Replica to fetch from.

--prefix Used to specify the beginning of a particular bundle UUID. Capitalized letters will be lower-cased as is done when users submit a uuid (all uuids have lower-cased letters upon ingestion into the dss). Characters other than letters, numbers, and dashes are not allowed and will error. The specified character(s) will return all available bundle uuids starting with that character(s).

--token Token to manage retries. End users constructing queries should not set this parameter.

--per-page Max number of results to return per page.

--search-after **Search-After-Context**. An internal state pointer parameter for use with pagination. This parameter is referenced by the `Link` header as described in the "Pagination" section. The API client should not need to set this parameter directly; it should instead directly fetch the URL given in the `Link` header.

--no-paginate Do not automatically page the responses

Default: True

get-bundles-checkout

Use this route with the `checkout_job_id` identifier returned by `POST /bundles/{uuid}/checkout`.

```
hca dss get-bundles-checkout [-h] --replica {aws,gcp} --checkout-job-id
                             CHECKOUT_JOB_ID
```

Named Arguments

--replica Possible choices: aws, gcp

Replica to fetch from.

--checkout-job-id A RFC4122-compliant ID for the checkout job request.

delete-bundle

Delete the bundle with the given UUID. This deletion is applied across replicas.

```
hca dss delete-bundle [-h] --reason REASON --uuid UUID --replica {aws,gcp}
                      [--version VERSION]
```

Named Arguments

--reason User-friendly reason for the bundle or timestamp-specfic bundle deletion.

--uuid A RFC4122-compliant ID for the bundle.

--replica Possible choices: aws, gcp

Replica to write to.

--version Timestamp of bundle creation in DSS_VERSION format.

get-bundle

Given a bundle UUID, return the latest version of that bundle. If the version is provided, that version of the bundle is returned instead.

```
hca dss get-bundle [-h] --uuid UUID [--version VERSION] --replica {aws,gcp}
                   [--directurls DIRECTURLS] [--presignedurls PRESIGNEDURLS]
                   [--token TOKEN] [--per-page PER_PAGE] [--start-at START_AT]
                   [--no-paginate]
```

Named Arguments

--uuid Bundle unique ID.

--version Timestamp of bundle creation in DSS_VERSION format.

--replica Possible choices: aws, gcp

Replica to fetch from.

--directurls When set to true, the response will contain API-specific URLs that are tied to the specified replica, for example `gs://bucket/object` or `s3://bucket/object`

This parameter is mutually exclusive with the presigned urls parameter. The use of presigned URLs is recommended for data access. Cloud native URLs are currently provided for a limited set of use cases and may not be provided in the future. If cloud native URLs are required, please contact the data store team regarding the credentials necessary to use them.

--presignedurls Include presigned URLs in the response. This is mutually exclusive with the directurls parameter.

--token Token to manage retries. End users constructing queries should not set this parameter.

--per-page Max number of results to return per page.

--start-at An internal state pointer parameter for use with pagination. This parameter is referenced by the `Link` header as described in the "Pagination" section. The API client should not need to set this parameter directly; it should instead directly fetch the URL given in the `Link` header.

--no-paginate Do not automatically page the responses

Default: True

patch-bundle

Add or remove files from a bundle. A specific version of the bundle to update must be provided, and a new version will be written. Bundle manifests exceeding 20,000 files will not be included in the Elasticsearch index document.

```
hca dss patch-bundle [-h] [--add-files ADD_FILES]
                     [--remove-files REMOVE_FILES] --uuid UUID --replica
                     {aws,gcp} --version VERSION
```

Named Arguments

--add-files List of new files to add to the bundle. File names must be unique.

--remove-files List of files to remove from the bundle. Files must match exactly to be removed. Files not found in the bundle are ignored.

--uuid A RFC4122-compliant ID of the bundle to update.

--replica Possible choices: aws, gcp

Replica to update the bundle on. Updates are propagated to other replicas.

--version Timestamp of the bundle to update in DSS_VERSION format format (required).

put-bundle

Create a new version of a bundle with a given UUID. The list of file UUID and versions to be included must be provided.

```
hca dss put-bundle [-h] --creator-uid CREATOR_UID --files FILES [FILES ...]
                   --uuid UUID --version VERSION --replica {aws,gcp}
```

Named Arguments

--creator-uid User ID who is creating this bundle.

--files This is a list of dictionaries describing each of the files. Each dictionary includes the fields: - The "uuid" of a file already previously uploaded with "PUT file/{uuid}". - The "version" timestamp of the file. - The "name" of the file. This can be most anything, and is the name the file will have when downloaded. - The "indexed" field, which specifies whether a file should be indexed or not. Bundle manifests exceeding 20,000 files will not be included in the Elasticsearch index document. Example representing 2 files with dummy values: [{'uuid': 'ce55fd51-7833-469b-be0b-5da88ebebfcd', 'version': '2017-06-16T193604.240704Z', 'name': 'dinosaur_dna.fa', 'indexed': False}, {'uuid': 'ae55fd51-7833-469b-be0b-5da88ebebfca', 'version': '0303-04-23T193604.240704Z', 'name': 'dragon_dna.fa', 'indexed': False}]

--uuid A RFC4122-compliant ID for the bundle.

--version Timestamp of bundle creation in DSS_VERSION format.

--replica Possible choices: aws, gcp

Replica to write to.

post-bundles-checkout

Initiate asynchronous checkout of a bundle. The response JSON contains a field, `checkout_job_id`, that can be used to query the status of the checkout via the `GET /bundles/checkout/{checkout_job_id}` API method. FIXME: document the error code returned when the bundle or specified version does not exist. TODO: After some time period, the data will be removed. TBD: This could be based on initial checkout time or last access time.

```
hca dss post-bundles-checkout [-h] [--destination DESTINATION] [--email EMAIL]
                              --uuid UUID [--version VERSION] --replica
                              {aws,gcp}
```

Named Arguments

--destination User-owned destination storage bucket.

--email An email address to send status updates to.

--uuid A RFC4122-compliant ID for the bundle.

--version Timestamp of file creation in DSS_VERSION format. If this is not provided, the latest version is returned.

--replica Possible choices: aws, gcp

Replica to fetch from.

get-collections

Return a list of a user's collections. Collections are sets of links to files, bundles, other collections, or fragments of JSON metadata files. Each entry in the input set of links is checked for referential integrity (the link target must exist in the replica referenced). Up to 1000 items can be referenced in a new collection, or added or removed using `PATCH /collections`. New collections are private to the authenticated user. Collection items are de-duplicated (if an identical item is given multiple times, it will only be added once). Collections are replicated across storage replicas similarly to files and bundles.

```
hca dss get-collections [-h] [--per-page PER_PAGE] [--start-at START_AT]
                        [--no-paginate]
```

Named Arguments

--per-page Max number of results to return per page.

--start-at An internal state pointer parameter for use with pagination. This parameter is referenced by the `Link` header as described in the "Pagination" section. The API client should not need to set this parameter directly; it should instead directly fetch the URL given in the `Link` header.

--no-paginate Do not automatically page the responses

Default: True

put-collection

Create a new collection. Collections are sets of links to files, bundles, other collections, or fragments of JSON metadata files. Each entry in the input set of links is checked for referential integrity (the link target must exist in the replica referenced). Up to 1000 items can be referenced in a new collection, or added or removed using `PATCH /collections`. New collections are private to the authenticated user. Collection items are de-duplicated (if an identical item is given multiple times, it will only be added once). Collections are replicated across storage replicas similarly to files and bundles.

```
hca dss put-collection [-h] --contents CONTENTS [CONTENTS ...] --description
                       DESCRIPTION --details DETAILS --name NAME --replica
                       {aws,gcp} --uuid UUID --version VERSION
```

Named Arguments

--contents A list of objects describing links to files, bundles, other collections, and metadata fragments that are part of the collection.

--description A long description of the collection, formatted in Markdown.

--details Supplementary JSON metadata for the collection.

--name A short name identifying the collection.

--replica Possible choices: aws, gcp

Replica to write to.

--uuid A RFC4122-compliant ID for the collection.

--version Timestamp of collection creation in DSS_VERSION format.

delete-collection

Delete a collection.

```
hca dss delete-collection [-h] --uuid UUID --replica {aws,gcp}
```

Named Arguments

--uuid A RFC4122-compliant ID for the collection.

--replica Possible choices: aws, gcp

Replica to delete from.

get-collection

Given a collection UUID, return the associated collection object.

```
hca dss get-collection [-h] --uuid UUID --replica {aws,gcp}
                       [--version VERSION]
```

Named Arguments

--uuid A RFC4122-compliant ID for the collection.

--replica Possible choices: aws, gcp

Replica to fetch from.

--version Timestamp of collection creation in DSS_VERSION format. If this is not provided, the latest version is returned.

patch-collection

Add or remove items from a collection. A specific version of the collection to update must be provided, and a new version will be written.

```
hca dss patch-collection [-h] [--add-contents ADD_CONTENTS]
                         [--description DESCRIPTION] [--details DETAILS]
                         [--name NAME] [--remove-contents REMOVE_CONTENTS]
                         --uuid UUID --replica {aws,gcp} --version VERSION
```

Named Arguments

--add-contents List of new items to add to the collection. Items are de-duplicated (if an identical item is already present in the collection or given multiple times, it will only be added once).

--description New description for the collection.

--details New details for the collection.

--name	New name for the collection.
--remove-contents	List of items to remove from the collection. Items must match exactly to be removed. Items not found in the collection are ignored.
--uuid	A RFC4122-compliant ID of the collection to update.
--replica	Possible choices: aws, gcp Replica to update the collection on. Updates are propagated to other replicas.
--version	Timestamp of the collection to update in DSS_VERSION format format (required).

get-events

Return urls where event data is available, with manifest of contents.

```
hca dss get-events [-h] [--from-date FROM_DATE] [--to-date TO_DATE] --replica
                   {aws,gcp} [--per-page PER_PAGE] [--token TOKEN]
                   [--no-paginate]
```

Named Arguments

--from-date	Timestamp to begin replaying events, in DSS_VERSION format. If this is not provided, replay from the earliest event.
--to-date	Timestamp to stop replaying events, in DSS_VERSION format. If this is not provided, replay to the latest event.
--replica	Possible choices: aws, gcp Replica to fetch from.
--per-page	Max number of results to return per page.
--token	Token to manage retries. End users constructing queries should not set this parameter.
--no-paginate	Do not automatically page the responses Default: True

get-event

Given a bundle UUID and version, return the bundle metadata document.

```
hca dss get-event [-h] --uuid UUID --version VERSION --replica {aws,gcp}
```

Named Arguments

--uuid	Bundle unique ID.
--version	Timestamp of bundle creation in DSS_VERSION format.

--replica Possible choices: aws, gcp

Replica to fetch from.

get-file

Given a file UUID, return the latest version of that file. If the version is provided, that version of the file is returned instead. Headers will contain the data store metadata for the file. This endpoint returns a HTTP redirect to another HTTP endpoint with the file contents.

```
hca dss get-file [-h] --uuid UUID --replica {aws,gcp} [--version VERSION]
                 [--token TOKEN] [--directurl DIRECTURL]
                 [--content-disposition CONTENT_DISPOSITION]
```

Named Arguments

--uuid A RFC4122-compliant ID for the file.

--replica Possible choices: aws, gcp

Replica to fetch from.

--version Timestamp of file creation in DSS_VERSION format. If this is not provided, the latest version is returned.

--token Token to manage retries. End users constructing queries should not set this parameter.

--directurl When set to true, the response will contain API-specific URLs that are tied to the specified replica, for example `gs://bucket/object` or `s3://bucket/object`

The use of presigned URLs is recommended for data access. Cloud native URLs are currently provided for a limited set of use cases and may not be provided in the future. If cloud native URLs are required, please contact the data store team regarding the credentials necessary to use them.

--content-disposition Optional and does not work when directurl=true (only works with the default presigned url response).

If this parameter is provided, the response from fetching the returned presigned url will include the specified Content-Disposition header.

This can be useful to indicate to a browser that a file should be downloaded rather than opened in a new tab, and can also supply the original filename in the response. Example:

```
content_disposition="attachment; filename=data.json"
```

head-file

Given a file UUID, return the metadata for the latest version of that file. If the version is provided, that version's metadata is returned instead. The metadata is returned in the headers. NOTE When using the HCA CLI, this will stream the file to stdout and may need to be piped. For example, `hca dss get-file --uuid UUID --replica aws > result.txt`

```
hca dss head-file [-h] --uuid UUID --replica {aws,gcp} [--version VERSION]
```

Named Arguments

--uuid	A RFC4122-compliant ID for the file.
--replica	Possible choices: aws, gcp Replica to fetch from.
--version	Timestamp of file creation in DSS_VERSION format. If this is not provided, the latest version is returned.

put-file

Create a new version of a file with a given UUID. The contents of the file are provided by the client by reference using a cloud object storage URL. The file on the cloud object storage service must have metadata set listing the file checksums and content-type. The metadata fields required are:

- hca-dss-sha256: SHA-256 checksum of the file
- hca-dss-sha1: SHA-1 checksum of the file
- hca-dss-s3_etag: S3 ETAG checksum of the file. See https://stackoverflow.com/q/12186993 for the general algorithm for how checksum is calculated. For files smaller than 64MB, this is the MD5 checksum of the file. For files larger than 64MB but smaller than 640,000MB, we use 64MB chunks. For files larger than 640,000MB, we use a chunk size equal to the total file size divided by 10000, rounded up to the nearest MB. MB, in this section, refers to 1,048,576 bytes. Note that 640,000MB is not the same as 640GB!
- hca-dss-crc32c: CRC-32C checksum of the file

```
hca dss put-file [-h] --creator-uid CREATOR_UID --source-url SOURCE_URL --uuid
                 UUID --version VERSION
```

Named Arguments

--creator-uid	User ID who is creating this file.
--source-url	Cloud bucket URL for source data. Example is "s3://bucket_name/serious_dna.fa" .
--uuid	A RFC4122-compliant ID for the file.
--version	Timestamp of file creation in DSS_VERSION format. If this is not provided, the latest version is returned.

post-search

Accepts Elasticsearch JSON query and returns matching bundle identifiers

Index design

The metadata seach index is implemented as a document-oriented database using Elasticsearch. The index stores all information relevant to a bundle within each bundle document, largely eliminating the need for object-relational mapping. This design is optimized for queries that filter the data.

To illustrate this concept, say our index stored information on three entities, `foo`, `bar`, and `baz`. A foo can have many bars and bars can have many bazes. If we were to index bazes in a document-oriented design, the information on the foo a bar comes from and the bazes it contains are combined into a single document. A example sketch of this is shown below in JSON-schema.

```
{
  "definitions": {
    "bar": {
      "type": "object",
      "properties": {
        "uuid": {
          "type": "string",
          "format": "uuid"
        },
        "foo": {
          "type": "object",
          "properties": {
            "uuid": {
              "type": "string",
              "format": "uuid"
            },
            ...
          }
        },
        "bazes": {
          "type": "array",
          "items": {
            "type": "string",
            "format": "uuid"
          }
        },
        ...
      }
    }
  }
}
```

This closely resembles the structure of DSS bundle documents: projects have many bundles and bundles have many files. Each bundle document is a concatenation of the metadata on the project it belongs to and the files it contains.

Limitations to index design

There are limitations to the design of DSS's metadata search index. A few important ones are listed below.

- Joins between bundle metadata must be conducted client-side
- Querying is schema-specific; fields or values changed between schema version will break queries that use those fields and values
- A new search index must be built for each schema version
- A lot of metadata is duplicated between documents

```
hca dss post-search [-h] --es-query ES_QUERY [--output-format {summary,raw}]
                    --replica {aws,gcp} [--per-page PER_PAGE]
                    [--search-after SEARCH_AFTER] [--no-paginate]
```

Named Arguments

--es-query Elasticsearch query

--output-format Possible choices: summary, raw

Specifies the output format. The default format, `summary`, is a list of UUIDs for bundles that match the query. Set this parameter to `raw` to get the verbatim JSON metadata for bundles that match the query. When using `output_format raw` the `per_page` size is limit to no more than 10 to avoid excessively large response sizes.

--replica Possible choices: aws, gcp

Replica to search.

--per-page Max number of results to return per page. When using `output_format raw` the `per_page` size is limit to no more than 10 to avoid excessively large response sizes.

--search-after **Search-After-Context**. An internal state pointer parameter for use with pagination. This parameter is referenced by the `Link` header as described in the "Pagination" section. The API client should not need to set this parameter directly; it should instead directly fetch the URL given in the `Link` header.

--no-paginate Do not automatically page the responses

Default: True

get-subscriptions

Return a list of associated subscriptions.

```
hca dss get-subscriptions [-h] --replica {aws,gcp}
                          [--subscription-type {elasticsearch,jmespath}]
```

Named Arguments

--replica Possible choices: aws, gcp

Replica to fetch from.

--subscription-type Possible choices: elasticsearch, jmespath

Type of subscriptions to fetch (elasticsearch or jmespath).

put-subscription

Register an HTTP endpoint that is to be notified when a given event occurs. Each user is allowed 100 subscriptions, a limit that may be increased in the future. Concerns about notification service limitations should be routed to the DSS development team.

```
hca dss put-subscription [-h] [--attachments ATTACHMENTS] --callback-url
                         CALLBACK_URL
                         [--encoding {application/json,multipart/form-data}]
                         [--es-query ES_QUERY] [--form-fields FORM_FIELDS]
                         [--hmac-key-id HMAC_KEY_ID]
                         [--hmac-secret-key HMAC_SECRET_KEY]
                         [--jmespath-query JMESPATH_QUERY]
                         [--method {POST,PUT}]
                         [--payload-form-field PAYLOAD_FORM_FIELD] --replica
                         {aws,gcp}
```

Named Arguments

--attachments The set of bundle metadata items to be included in the payload of a notification request to a subscription endpoint. Each property in this object represents an attachment to the notification payload. Each attachment will be a child property of the `attachments` property of the payload. The name of such a child property can be chosen freely provided it does not start with an underscore. For example, if the subscription is

```
{
  "attachments": {
    "taxon": {
      "type": "jmespath",
      "expression": "files.biomaterial_json.biomaterials[].
↪content.biomaterial_core.ncbi_taxon_id[]"
    }
  }
}
```

the corresponding notification payload will contain the following entry

```
"attachments": {
  "taxon": [9606, 9606]
}
```

If a general error occurs during the processing of attachments, the notification will be sent with `attachments` containing only the reserved `_errors` attachment containing a string describing the error. If an error occurs during the processing of a specific attachment, the notification will be sent with all successfully processed attachments and additionally the `_errors` attachment containing an object with one property for each failed attachment. For example,

```
"attachments": {
  "taxon": [9606, 9606]
  "_errors" {
    "biomaterial": "Some error occurred"
  }
}
```

The value of the `attachments` property must be less than or equal to 128 KiB in size when serialized to JSON and encoded as UTF-8. If it is not, the notification will be sent with “attachments”: { “_errors”: “Attachments too large (131073 bytes)” }

--callback-url The subscriber's URL. An HTTP request is made to the specified URL for every attempt to deliver a notification to the subscriber. If the HTTP response code is 2XX, the delivery attempt is considered successful. Otherwise, more attempts will be made with an exponentially increasing delay between attempts, until an attempt is successful or the a maximum number of attempts is reached. Occasionally, duplicate notifications may be sent. It is up to the receiver of the notification to tolerate duplicate notifications.

--encoding Possible choices: application/json, multipart/form-data

The MIME type describing the encoding of the request body * `application/json` - the HTTP request body is the notification payload as JSON * `multipart/form-data` - the HTTP request body is a list of form fields, each consisting of a name and a corresponding value. See https://tools.ietf.org/html/rfc7578 for details on this encoding. The actual notification payload will be placed as JSON into a field of the name specified via `payload_form_field`.

--es-query An Elasticsearch query for restricting the set of bundles for which the subscriber is notified. The subscriber will only be notified for newly indexed bundles that match the given query. If this parameter is present the subscription will be of type `elasticsearch`, otherwise it will be of type `jmespath`.

--form-fields A collection of static form fields to be supplied in the request body, alongside the actual notification payload. The value of each field must be a string. For example, if the subscriptions has this property set to `{"foo" : "bar"}`, the corresponding notification HTTP request body will consist of a multipart frame with two frames,

```
----------------2769baffc4f24cbc83ced26aa0c2f712
Content-Disposition: form-data; name="foo"
bar
----------------2769baffc4f24cbc83ced26aa0c2f712
Content-Disposition: form-data; name="payload"
{"transaction_id": "301c9079-3b20-4311-a131-bcda9b7f08ba",
↪"subscription_id": ...
```

Since the type of this property is `object`, multi-valued fields are not supported. This property is ignored unless `encoding` is `multipart/form-data`.

--hmac-key-id An optional key ID to use with `hmac_secret_key`.

--hmac-secret-key The key for signing requests to the subscriber's URL. The signature will be constructed according to https://tools.ietf.org/html/draft-cavage-http-signatures and transmitted in the HTTP `Authorization` header.

--jmespath-query An JMESPath query for restricting the set of bundles for which the subscriber is notified. The subscriber will only be notified for new bundles that match the given query. If `es_query` is specified, the subscription will be of type `elasticsearch`. If `es_query` is not present, the subscription will be of type `jmespath`

--method Possible choices: POST, PUT

The HTTP request method to use when delivering a notification to the subscriber.

--payload-form-field The name of the form field that will hold the notification payload when the request is made. If the default name of the payload field collides with that of a field in `form_fields`, this porperty can be used to rename the payload and

avoid the collision. This property is ignored unless `encoding` is `multipart/form-data`.

--replica Possible choices: aws, gcp

Replica to write to.

delete-subscription

Delete a registered event subscription. The associated query will no longer trigger a callback if a matching document is added to the system.

```
hca dss delete-subscription [-h] --uuid UUID --replica {aws,gcp}
                            [--subscription-type {elasticsearch,jmespath}]
```

Named Arguments

--uuid A RFC4122-compliant ID for the subscription.

--replica Possible choices: aws, gcp

Replica to delete from.

--subscription-type Possible choices: elasticsearch, jmespath

type of subscriptions to fetch (elasticsearch or jmespath)

get-subscription

Given a subscription UUID, return the associated subscription.

```
hca dss get-subscription [-h] --uuid UUID --replica {aws,gcp}
                         [--subscription-type {elasticsearch,jmespath}]
```

Named Arguments

--uuid A RFC4122-compliant ID for the subscription.

--replica Possible choices: aws, gcp

Replica to fetch from.

--subscription-type Possible choices: elasticsearch, jmespath

type of subscriptions to fetch (elasticsearch or jmespath)

login

This command may open a browser window to ask for your consent to use web service authentication credentials.

Use –remote if using the CLI in a remote environment

```
hca dss login [-h] [--access-token ACCESS_TOKEN] [--remote]
```

Named Arguments

--access-token Default: “”

--remote Default: False

logout

Clear sphinx-build dss authentication credentials previously configured with sphinx-build dss login.

```
hca dss logout [-h]
```

upload

Upload a directory of files from the local filesystem and create a bundle containing the uploaded files. This method requires the use of a client-controlled object storage bucket to stage the data for upload.

```
hca dss upload [-h] --src-dir SRC_DIR --replica REPLICA --staging-bucket
               STAGING_BUCKET [--timeout-seconds TIMEOUT_SECONDS]
               [--no-progress] [--bundle-uuid BUNDLE_UUID]
```

Named Arguments

--src-dir file path to a directory of files to upload to the replica.

--replica the replica to upload to. The supported replicas are: aws for Amazon Web Services, and gcp for Google Cloud Platform. [aws, gcp]

--staging-bucket a client controlled AWS S3 storage bucket to upload from.

--timeout-seconds the time to wait for a file to upload to replica.

Default: 1200

--no-progress if set, will not report upload progress. Note that even if this flag is not set, progress will not be reported if the logging level is higher than INFO or if the session is not interactive.

Default: False

--bundle-uuid

download

Download a bundle and save it to the local filesystem as a directory.

By default, all data and metadata files are downloaded. To disable the downloading of data, use the –no-data flag if using the CLI or pass the no_data=True argument if calling the download() API method. Likewise, to disable the downloading of metadata, use the –no-metadata flag for the CLI or pass the no_metadata=True argument if calling the download() API method.

If a retryable exception occurs, we wait a bit and retry again. The delay increases each time we fail and decreases each time we successfully read a block. We set a quota for the number of failures that goes up with every successful block read and down with each failure.

```
hca dss download [-h] --bundle-uuid BUNDLE_UUID --replica REPLICA
                 [--version VERSION] [--download-dir DOWNLOAD_DIR]
                 [--metadata-filter METADATA_FILTER [METADATA_FILTER ...]]
                 [--data-filter DATA_FILTER [DATA_FILTER ...]] [--no-metadata]
                 [--no-data] [--num-retries NUM_RETRIES]
                 [--min-delay-seconds MIN_DELAY_SECONDS]
```

Named Arguments

--bundle-uuid The uuid of the bundle to download

--replica the replica to download from. The supported replicas are: aws for Amazon Web Services, and gcp for Google Cloud Platform. [aws, gcp]

--version The version to download, else if not specified, download the latest. The version is a timestamp of bundle creation in RFC3339

Default: “”

--download-dir The directory into which to download

Default: “”

--metadata-filter One or more shell patterns against which all metadata files in the bundle will be matched case-sensitively. A file is considered a metadata file if the indexed property in the manifest is set. If and only if a metadata file matches any of the patterns in metadata_files will it be downloaded.

Default: (‘*’,)

--data-filter One or more shell patterns against which all data files in the bundle will be matched case-sensitively. A file is considered a data file if the indexed property in the manifest is not set. The file will be downloaded only if a data file matches any of the patterns in data_files will it be downloaded.

Default: (‘*’,)

--no-metadata Exclude metadata files. Cannot be set when –metadata-filter is also set.

Default: False

--no-data Exclude data files. Cannot be set when –data-filter is also set.

Default: False

--num-retries The initial quota of download failures to accept before exiting due to failures. The number of retries increase and decrease as file chucks succeed and fail.

Default: 10

--min-delay-seconds The minimum number of seconds to wait in between retries.

Default: 0.25

download-manifest

Files are always downloaded to a cache / filestore directory called ‘.hca’. This directory is created in the current directory where download is initiated. A copy of the manifest used is also written to the current directory. This manifest has an added column that lists the paths of the files within the ‘.hca’ filestore.

The default layout is none. In this layout all of the files are downloaded to the filestore and the recommended way of accessing the files in by parsing the manifest copy that's written to the download directory.

The bundle layout still downloads all of files to the filestore. For each bundle mentioned in the manifest a directory is created. All relevant metadata files for each bundle are linked into these directories in addition to relevant data files mentioned in the manifest.

Each row in the manifest represents one file in DSS. The manifest must have a header row. The header row must declare the following columns:

- bundle_uuid - the UUID of the bundle containing the file in DSS.
- bundle_version - the version of the bundle containing the file in DSS.
- file_name - the name of the file as specified in the bundle.
- file_uuid - the UUID of the file in the DSS.
- file_sha256 - the SHA-256 hash of the file.
- file_size - the size of the file.

The TSV may have additional columns. Those columns will be ignored. The ordering of the columns is insignificant because the TSV is required to have a header row.

This download format will serve as the main storage format for downloaded files. If a user specifies a different format for download (coming in the future) the files will first be downloaded in this format, then hard-linked to the user's preferred format.

```
hca dss download-manifest [-h] --manifest MANIFEST --replica REPLICA
                          [--layout LAYOUT] [--no-metadata] [--no-data]
                          [--num-retries NUM_RETRIES]
                          [--min-delay-seconds MIN_DELAY_SECONDS]
                          [--download-dir DOWNLOAD_DIR]
```

Named Arguments

--manifest	The path to a TSV (tab-separated values) file listing files to download. If the directory for download already contains the manifest, the manifest will be overwritten to include a column with paths into the filestore.
--replica	The replica from which to download. The supported replicas are: aws for Amazon Web Services, and gcp for Google Cloud Platform. [aws, gcp]
--layout	The layout of the downloaded files. Currently two options are supported, 'none' (the default), and 'bundle'. Default: "none"
--no-metadata	Exclude metadata files. Cannot be set when –metadata-filter is also set. Default: False
--no-data	Exclude data files. Cannot be set when –data-filter is also set. Default: False
--num-retries	The initial quota of download failures to accept before exiting due to failures. The number of retries increase and decrease as file chucks succeed and fail. Default: 10

--min-delay-seconds The minimum number of seconds to wait in between retries for downloading any file

Default: 0.25

--download-dir The directory into which to download

Default: “”

create-version

Prints a timestamp that can be used for versioning

```
hca dss create-version [-h]
```

download-collection

Download a bundle and save it to the local filesystem as a directory.

```
hca dss download-collection [-h] --uuid UUID --replica REPLICA
                            [--version VERSION] [--download-dir DOWNLOAD_DIR]
```

Named Arguments

--uuid The uuid of the collection to download

--replica the replica to download from. The supported replicas are: aws for Amazon Web Services, and gcp for Google Cloud Platform. [aws, gcp]

--version The version to download, else if not specified, download the latest. The version is a timestamp of bundle creation in RFC3339

--download-dir The directory into which to download

Default: “”

6.1.2.5 auth

Interact with the HCA authorization and authentication system.

```
hca auth [-h]
         {get-login,get-logout,get-openid-configuration,get-jwks.json,get-oauth-
↪authorize,post-oauth-revoke,post-oauth-token,get-oauth-userinfo,post-oauth-userinfo,
↪get-echo,post-v1-policies-evaluate,get-v1-users,post-v1-user,get-v1-user,put-v1-
↪user,get-v1-user-owns,get-v1-user-groups,put-v1-user-group,get-v1-user-roles,put-v1-
↪user-role,put-v1-user-policy,get-v1-groups,post-v1-group,get-v1-group,delete-v1-
↪group,get-v1-group-roles,put-v1-group-role,get-v1-group-users,put-v1-group-user,put-
↪v1-group-policy,get-v1-roles,post-v1-role,get-v1-role,delete-v1-role,put-v1-role-
↪policy,login,logout}
         ...
```

6.1.2.5.1 Sub-commands:

get-login

Send the user agent to an identity provider selector and generate a user account to establish the user's identity. This is a redirect endpoint.

```
hca auth get-login [-h] --redirect-uri REDIRECT_URI [--state STATE]
```

Named Arguments

--redirect-uri Where to redirect to once login is complete.

--state An opaque parameter that is returned back to the `redirect_uri`.

get-logout

Logout the user from current sessions with the OIDC provider. You can log the users out from a specific application if the you know the client_id for the application. Otherwise the user will be logged out of the default application by oauth2_config.

```
hca auth get-logout [-h] [--client-id CLIENT_ID]
```

Named Arguments

--client-id

get-openid-configuration

This endpoint is part of OIDC, see documentation at Provider Config

```
hca auth get-openid-configuration [-h] --host HOST
```

Named Arguments

--host Must be `auth.data.humancellatlas.org`.

get-jwks.json

Provide the public key used to sign all JWTs minted by the OIDC provider. See JSON Web Key Set for more info.

```
hca auth get-jwks.json [-h]
```

get-oauth-authorize

This endpoint is part of OIDC and is used to redirect to an openid provider. See Auth Request

```
hca auth get-oauth-authorize [-h] [--redirect-uri REDIRECT_URI]
                             [--state STATE] [--client-id CLIENT_ID]
                             [--scope SCOPE] [--respone-type RESPONE_TYPE]
                             [--nonce NONCE] [--prompt PROMPT]
```

Named Arguments

--redirect-uri

--state

--client-id

--scope

--respone-type

--nonce

--prompt

post-oauth-revoke

Revokes a refresh token from a client making all future token refresh requests fail.

```
hca auth post-oauth-revoke [-h] --client-id CLIENT_ID --token TOKEN
```

Named Arguments

--client-id

--token The refresh token to revoke.

post-oauth-token

This endpoint is part of OIDC and is used to redirect to an openid provider. See Token Endpoint, and Refresh Tokens

```
hca auth post-oauth-token [-h]
```

get-oauth-userinfo

This endpoint is part of OIDC and is used to redirect to an openid provider. See User Info

```
hca auth get-oauth-userinfo [-h]
```

post-oauth-userinfo

This endpoint is part of OIDC and is used to redirect to an openid provider. See User Info

```
hca auth post-oauth-userinfo [-h]
```

get-echo

Echoes the response back.

```
hca auth get-echo [-h]
```

post-v1-policies-evaluate

Given a set of principals, actions, and resources, return a set of access control decisions.

```
hca auth post-v1-policies-evaluate [-h] --principal PRINCIPAL --action ACTION
                                   [ACTION ...] --resource RESOURCE
                                   [RESOURCE ...]
```

Named Arguments

--principal Attested user identifier.

--action The action the principal is attempting to perform.

--resource The resource the principal will perform the action against.

get-v1-users

Paginate through all users.

```
hca auth get-v1-users [-h] [--next-token NEXT_TOKEN] [--per-page PER_PAGE]
                      [--no-paginate]
```

Named Arguments

--next-token

--per-page

--no-paginate Do not automatically page the responses

Default: True

post-v1-user

Create a new user with the specified groups, roles, and iam policy.

```
hca auth post-v1-user [-h] --user-id USER_ID [--groups GROUPS] [--roles ROLES]
                      [--policy POLICY]
```

Named Arguments

--user-id Used to identify users, groups, and roles.

--groups

--roles

--policy

get-v1-user

Retrieve information about the user's status and the policies attached.

```
hca auth get-v1-user [-h] --user-id USER_ID
```

Named Arguments

--user-id User ID (email).

put-v1-user

Enable or disable a user. A disabled user will return false for all evaluations with that user as principal.

```
hca auth put-v1-user [-h] --user-id USER_ID --status STATUS
```

Named Arguments

--user-id User ID (email).

--status

get-v1-user-owns

Paginate through a list of resources owned by a user.

```
hca auth get-v1-user-owns [-h] --user-id USER_ID [--next-token NEXT_TOKEN]
                          [--per-page PER_PAGE] --resource-type RESOURCE_TYPE
                          [--no-paginate]
```

Named Arguments

--user-id User ID (email).

--next-token

--per-page

--resource-type

--no-paginate Do not automatically page the responses

Default: True

get-v1-user-groups

Paginate through a list of groups of which a user is a member.

```
hca auth get-v1-user-groups [-h] --user-id USER_ID [--next-token NEXT_TOKEN]
                            [--per-page PER_PAGE] [--no-paginate]
```

Named Arguments

--user-id User ID (email).

--next-token

--per-page

--no-paginate Do not automatically page the responses

Default: True

put-v1-user-group

Modify group(s) in which a user is a member.

```
hca auth put-v1-user-group [-h] [--groups GROUPS] --user-id USER_ID --action
                           ACTION
```

Named Arguments

--groups

--user-id User ID (email).

--action

get-v1-user-roles

Paginate through all roles attached to a user.

```
hca auth get-v1-user-roles [-h] --user-id USER_ID [--next-token NEXT_TOKEN]
                           [--per-page PER_PAGE] [--no-paginate]
```

Named Arguments

--user-id User ID (email).

--next-token

--per-page

--no-paginate Do not automatically page the responses

Default: True

put-v1-user-role

Modify the role(s) attached to a user.

```
hca auth put-v1-user-role [-h] [--roles ROLES] --user-id USER_ID --action
                          ACTION
```

Named Arguments

--roles

--user-id User ID (email).

--action

put-v1-user-policy

Modify or add the user's IAM policy.

```
hca auth put-v1-user-policy [-h] [--policy POLICY] --user-id USER_ID
```

Named Arguments

--policy

--user-id User ID (email).

get-v1-groups

Paginate through all groups.

```
hca auth get-v1-groups [-h] [--next-token NEXT_TOKEN] [--per-page PER_PAGE]
                       [--no-paginate]
```

Named Arguments

--next-token

--per-page

--no-paginate Do not automatically page the responses

Default: True

post-v1-group

Create a new group, attach an IAM policy, and assign roles.

```
hca auth post-v1-group [-h] --group-id GROUP_ID [--policy POLICY]
                       [--roles ROLES]
```

Named Arguments

--group-id Used to identify users, groups, and roles.

--policy

--roles

get-v1-group

Get properties of a group, including the group's IAM policy.

```
hca auth get-v1-group [-h] --group-id GROUP_ID
```

Named Arguments

--group-id The name of the group.

delete-v1-group

Remove all users, policies, and roles from the group, and delete the group.

```
hca auth delete-v1-group [-h] --group-id GROUP_ID
```

Named Arguments

--group-id The name of the group.

get-v1-group-roles

Paginate through all roles assigned to the group.

```
hca auth get-v1-group-roles [-h] --group-id GROUP_ID [--next-token NEXT_TOKEN]
                            [--per-page PER_PAGE] [--no-paginate]
```

Named Arguments

--group-id The name of the group.

--next-token

--per-page

--no-paginate Do not automatically page the responses

Default: True

put-v1-group-role

Modify the role(s) assigned to a group.

```
hca auth put-v1-group-role [-h] [--roles ROLES] --group-id GROUP_ID --action
                           ACTION
```

Named Arguments

--roles

--group-id The name of the group.

--action

get-v1-group-users

Paginate through all users in a group.

```
hca auth get-v1-group-users [-h] --group-id GROUP_ID [--next-token NEXT_TOKEN]
                            [--per-page PER_PAGE] [--no-paginate]
```

Named Arguments

--group-id The name of the group.

--next-token

--per-page

--no-paginate Do not automatically page the responses

Default: True

put-v1-group-user

Modify the user(s) assigned to a group.

```
hca auth put-v1-group-user [-h] [--users USERS] --group-id GROUP_ID --action
                           ACTION
```

Named Arguments

--users

--group-id The name of the group.

--action

put-v1-group-policy

Modify or create a policy attached to a group.

```
hca auth put-v1-group-policy [-h] [--policy POLICY] --group-id GROUP_ID
```

Named Arguments

--policy

--group-id The name of the group.

get-v1-roles

Paginate through all roles.

```
hca auth get-v1-roles [-h] [--next-token NEXT_TOKEN] [--per-page PER_PAGE]
                      [--no-paginate]
```

Named Arguments

--next-token

--per-page

--no-paginate Do not automatically page the responses

Default: True

post-v1-role

Create a new role and attach a IAM policy.

```
hca auth post-v1-role [-h] --role-id ROLE_ID --policy POLICY
```

Named Arguments

--role-id Used to identify users, groups, and roles.

--policy

get-v1-role

Get properties of a role.

```
hca auth get-v1-role [-h] --role-id ROLE_ID
```

Named Arguments

--role-id The name of the role.

delete-v1-role

Remove the role from all users and groups, and finally delete the role.

```
hca auth delete-v1-role [-h] --role-id ROLE_ID
```

Named Arguments

--role-id The name of the role.

put-v1-role-policy

Modify the IAM policy attached to the role.

```
hca auth put-v1-role-policy [-h] [--policy POLICY] --role-id ROLE_ID
```

Named Arguments

--policy

--role-id The name of the role.

login

This command may open a browser window to ask for your consent to use web service authentication credentials.

Use –remote if using the CLI in a remote environment

```
hca auth login [-h] [--access-token ACCESS_TOKEN] [--remote]
```

Named Arguments

--access-token Default: “”

--remote Default: False

logout

Clear sphinx-build auth authentication credentials previously configured with sphinx-build auth login.

```
hca auth logout [-h]
```

Links: genindex / modindex / search

6.2 API documentation

6.2.1 Data Storage System

class hca.dss.**DSSClient**(**args*, ***kwargs*)

Human Cell Atlas Data Coordination Platform Data Storage System API

The DSS API requires clients to follow certain HTTP protocol semantics that may require extra configuration in your HTTP client. The reference CLI and SDK (https://hca.readthedocs.io/) is pre-configured to do this. If writing your own client, please note the following:

301 redirects: Some DSS API routes may return one or more HTTP 301 redirects, including potentially redirects to themselves (combined with the **Retry-After** delay described below). The client must follow these redirects to obtain the resource requested.

Retry-After header: Some DSS API routes may use the **Retry-After** header in combination with HTTP 301 or 500 series response codes. The client must follow the HTTP specification and wait the designated time period before continuing with the next request.

General retry logic: If you are building an application that will issue high numbers of API requests, you should be prepared for the possibility that a small fraction of requests fails due to network or server errors. In these situations, the HTTP client should follow best practice HTTP retry semantics. For example, clients may be configured to retry 5 times while waiting for an exponential number of seconds (1, 2, 4, 8, 16 seconds) upon encountering any 500 series response code, connect or read timeout.

The following Python code demonstrates an example configuration of the popular Requests library per the above guidance:

```
import requests, requests.packages.urllib3.util.retry
class RetryPolicy(requests.packages.urllib3.util.retry.Retry):
    def __init__(self, retry_after_status_codes={301}, **kwargs):
        super(RetryPolicy, self).__init__(**kwargs)
        self.RETRY_AFTER_STATUS_CODES = frozenset(retry_after_status_codes |␣
↪retry.Retry.RETRY_AFTER_STATUS_CODES)

retry_policy = RetryPolicy(read=5, status=5, status_forcelist=frozenset({500, 502,
↪ 503, 504}))
s = requests.Session()
a = requests.adapters.HTTPAdapter(max_retries=retry_policy)
s.mount('https://', a)
print(s.get("https://dss.data.humancellatlas.org").content)
```

DSS supports webhook subscriptions for data events like bundle creation and deletion. Webhooks are callbacks to a public HTTPS endpoint provided by your application. When an event matching your subscription occurs, DSS will send a push notification (via an HTTPS POST or PUT request), giving your application an up-to-date stream of system activity. Subscriptions are delivered with the payload format

```
{
  'transaction_id': {uuid},
  'subscription_id': {uuid},
  'event_type': "CREATE"|"TOMBSTONE"|"DELETE",  # JMESPath subscriptions only
  'match': {
    'bundle_uuid': {uuid},
    'bundle_version': {version},
  }
  'jmespath_query': {jmespath_query},  # JMESPath subscriptions only
  'es_query': {es_query},  # Elasticsearch subscriptions only
  'attachments': {
    "attachment_name_1": {value},
    "attachment_name_1": {value},
    ...
    "_errors": [...]
  }
}
```

DSS_VERSION: a timestamp that generally follows RFC3339 format guide. However there are a few differences. DSS_VERSION must always be in UTC time, ':' are removed from the time, and the fractional seconds extends to 6 decimal places. Using the first example found here, the RFC3339 version would be `1985-04-12T23:20:50.52Z` while the DSS_VERSION would be `1985-04-12T232050.520000Z`

The DSS API supports pagination in a manner consistent with the GitHub API, which is based on RFC 5988. When the results of an API call exceed the page size specified, the HTTP response will contain a `Link` header of the following form: `Link: <https://dss.data.humancellatlas.org/v1/search?replica=aws&per_page=100&search_after=123>; rel="next"`. The URL in the header refers to the next page of the results to be fetched; if no `Link rel="next"` URL is included, then all results have been fetched. The client should recognize and parse the `Link` header appropriately according to RFC 5988, and retrieve the next page if requested by the user, or if all results are being retrieved.

clear_cache()
Clear the cached API definitions for a component. This can help resolve errors communicating with the API.

create_version()
Prints a timestamp that can be used for versioning

classmethod delete_bundle(*client*, *reason: str = None*, *uuid: str = None*, *replica: str = None*, *version: Optional[str] = None*)
Delete a bundle or a specific bundle version

Parameters

- **reason** (*<class 'str'>*) – User-friendly reason for the bundle or timestamp-specfic bundle deletion.
- **uuid** (*<class 'str'>*) – A RFC4122-compliant ID for the bundle.
- **replica** (*<class 'str'>*) – Replica to write to.
- **version** (*typing.Union[str, NoneType]*) – Timestamp of bundle creation in DSS_VERSION format.

Delete the bundle with the given UUID. This deletion is applied across replicas.

classmethod delete_collection(*client*, *uuid: str = None*, *replica: str = None*)
Delete a collection.

Parameters

- **uuid** (*<class 'str'>*) – A RFC4122-compliant ID for the collection.
- **replica** (*<class 'str'>*) – Replica to delete from.

Delete a collection.

classmethod delete_subscription(*client*, *uuid: str = None*, *replica: str = None*, *subscription_type: Optional[str] = 'jmespath'*)

Delete an event subscription.

Parameters

- **uuid** (*<class 'str'>*) – A RFC4122-compliant ID for the subscription.
- **replica** (*<class 'str'>*) – Replica to delete from.
- **subscription_type** (*typing.Union[str, NoneType]*) – type of subscriptions to fetch (elasticsearch or jmespath)

Delete a registered event subscription. The associated query will no longer trigger a callback if a matching document is added to the system.

download(*bundle_uuid*, *replica*, *version=''*, *download_dir=''*, *metadata_filter=('*',)*, *data_filter=('*',)*, *no_metadata=False*, *no_data=False*, *num_retries=10*, *min_delay_seconds=0.25*)

Download a bundle and save it to the local filesystem as a directory.

Parameters

- **bundle_uuid** (*str*) – The uuid of the bundle to download
- **replica** (*str*) – the replica to download from. The supported replicas are: *aws* for Amazon Web Services, and *gcp* for Google Cloud Platform. [aws, gcp]
- **version** (*str*) – The version to download, else if not specified, download the latest. The version is a timestamp of bundle creation in RFC3339
- **download_dir** (*str*) – The directory into which to download
- **metadata_filter** (*iterable*) – One or more shell patterns against which all metadata files in the bundle will be matched case-sensitively. A file is considered a metadata file if the *indexed* property in the manifest is set. If and only if a metadata file matches any of the patterns in *metadata_files* will it be downloaded.
- **data_filter** (*iterable*) – One or more shell patterns against which all data files in the bundle will be matched case-sensitively. A file is considered a data file if the *indexed* property in the manifest is not set. The file will be downloaded only if a data file matches any of the patterns in *data_files* will it be downloaded.
- **no_metadata** – Exclude metadata files. Cannot be set when –metadata-filter is also set.
- **no_data** – Exclude data files. Cannot be set when –data-filter is also set.
- **num_retries** (*int*) – The initial quota of download failures to accept before exiting due to failures. The number of retries increase and decrease as file chucks succeed and fail.
- **min_delay_seconds** (*float*) – The minimum number of seconds to wait in between retries.

Download a bundle and save it to the local filesystem as a directory.

By default, all data and metadata files are downloaded. To disable the downloading of data, use the *–no-data* flag if using the CLI or pass the *no_data=True* argument if calling the *download()* API method. Likewise, to disable the downloading of metadata, use the *–no-metadata* flag for the CLI or pass the *no_metadata=True* argument if calling the *download()* API method.

If a retryable exception occurs, we wait a bit and retry again. The delay increases each time we fail and decreases each time we successfully read a block. We set a quota for the number of failures that goes up with every successful block read and down with each failure.

download_collection(*uuid*, *replica*, *version=None*, *download_dir=''*)
Download a bundle and save it to the local filesystem as a directory.

Parameters

- **uuid** (*str*) – The uuid of the collection to download
- **replica** (*str*) – the replica to download from. The supported replicas are: *aws* for Amazon Web Services, and *gcp* for Google Cloud Platform. [aws, gcp]
- **version** (*str*) – The version to download, else if not specified, download the latest. The version is a timestamp of bundle creation in RFC3339
- **download_dir** (*str*) – The directory into which to download

Download a bundle and save it to the local filesystem as a directory.

download_manifest(*manifest*, *replica*, *layout='none'*, *no_metadata=False*, *no_data=False*, *num_retries=10*, *min_delay_seconds=0.25*, *download_dir=''*)
Process the given manifest file in TSV (tab-separated values) format and download the files referenced by it.

Parameters

- **layout** (*str*) – The layout of the downloaded files. Currently two options are supported, 'none' (the default), and 'bundle'.
- **manifest** (*str*) – The path to a TSV (tab-separated values) file listing files to download. If the directory for download already contains the manifest, the manifest will be overwritten to include a column with paths into the filestore.
- **replica** (*str*) – The replica from which to download. The supported replicas are: *aws* for Amazon Web Services, and *gcp* for Google Cloud Platform. [aws, gcp]
- **no_metadata** – Exclude metadata files. Cannot be set when –metadata-filter is also set.
- **no_data** – Exclude data files. Cannot be set when –data-filter is also set.
- **num_retries** (*int*) – The initial quota of download failures to accept before exiting due to failures. The number of retries increase and decrease as file chucks succeed and fail.
- **min_delay_seconds** (*float*) – The minimum number of seconds to wait in between retries for downloading any file
- **download_dir** (*str*) – The directory into which to download

Files are always downloaded to a cache / filestore directory called '.hca'. This directory is created in the current directory where download is initiated. A copy of the manifest used is also written to the current directory. This manifest has an added column that lists the paths of the files within the '.hca' filestore.

The default layout is **none**. In this layout all of the files are downloaded to the filestore and the recommended way of accessing the files in by parsing the manifest copy that's written to the download directory.

The bundle layout still downloads all of files to the filestore. For each bundle mentioned in the manifest a directory is created. All relevant metadata files for each bundle are linked into these directories in addition to relevant data files mentioned in the manifest.

Each row in the manifest represents one file in DSS. The manifest must have a header row. The header row must declare the following columns:

- *bundle_uuid* - the UUID of the bundle containing the file in DSS.
- *bundle_version* - the version of the bundle containing the file in DSS.
- *file_name* - the name of the file as specified in the bundle.
- *file_uuid* - the UUID of the file in the DSS.
- *file_sha256* - the SHA-256 hash of the file.
- *file_size* - the size of the file.

The TSV may have additional columns. Those columns will be ignored. The ordering of the columns is insignificant because the TSV is required to have a header row.

This download format will serve as the main storage format for downloaded files. If a user specifies a different format for download (coming in the future) the files will first be downloaded in this format, then hard-linked to the user's preferred format.

expired_token()
Return True if we have an active session containing an expired (or nearly expired) token.

classmethod get_bundle(*client*, *uuid: str = None*, *version: Optional[str] = None*, *replica: str = None*, *directurls: Optional[str] = None*, *presignedurls: Optional[str] = None*, *token: Optional[str] = None*, *per_page: Optional[str] = 500*, *start_at: Optional[str] = None*)
Retrieve a bundle given a UUID and optionally a version.

Pagination

This method supports pagination. Use `DSSClient.get_bundle.iterate(**kwargs)` to create a generator that yields all results, making multiple requests over the wire if necessary:

```
for result in DSSClient.get_bundle.iterate(**kwargs):
    ...
```

The keyword arguments for `DSSClient.get_bundle.iterate()` are identical to the arguments for `DSSClient.get_bundle()` listed here.

Parameters

- **uuid** (*<class 'str'>*) – Bundle unique ID.
- **version** (*typing.Union[str, NoneType]*) – Timestamp of bundle creation in DSS_VERSION format.
- **replica** (*<class 'str'>*) – Replica to fetch from.
- **directurls** (*typing.Union[str, NoneType]*) – When set to true, the response will contain API-specific URLs that are tied to the specified replica, for example `gs://bucket/object` or `s3://bucket/object` This parameter is mutually exclusive with the presigned urls parameter. The use of presigned URLs is recommended for data access. Cloud native URLs are currently provided for a limited set of use cases and may not be provided in the future. If cloud native URLs are required, please contact the data store team regarding the credentials necessary to use them.
- **presignedurls** (*typing.Union[str, NoneType]*) – Include presigned URLs in the response. This is mutually exclusive with the directurls parameter.
- **token** (*typing.Union[str, NoneType]*) – Token to manage retries. End users constructing queries should not set this parameter.

- **per_page** (*typing.Union[str, NoneType]*) – Max number of results to return per page.
- **start_at** (*typing.Union[str, NoneType]*) – An internal state pointer parameter for use with pagination. This parameter is referenced by the `Link` header as described in the "Pagination" section. The API client should not need to set this parameter directly; it should instead directly fetch the URL given in the `Link` header.

Given a bundle UUID, return the latest version of that bundle. If the version is provided, that version of the bundle is returned instead.

classmethod get_bundles_all(*client*, *replica: str = None*, *prefix: Optional[str] = None*, *token: Optional[str] = None*, *per_page: Optional[str] = 100*, *search_after: Optional[str] = None*)

List through all available bundles.

Pagination

This method supports pagination. Use `DSSClient.get_bundles_all.iterate(**kwargs)` to create a generator that yields all results, making multiple requests over the wire if necessary:

```
for result in DSSClient.get_bundles_all.iterate(**kwargs):
    ...
```

The keyword arguments for `DSSClient.get_bundles_all.iterate()` are identical to the arguments for `DSSClient.get_bundles_all()` listed here.

Parameters

- **replica** (*<class 'str'>*) – Replica to fetch from.
- **prefix** (*typing.Union[str, NoneType]*) – Used to specify the beginning of a particular bundle UUID. Capitalized letters will be lower-cased as is done when users submit a uuid (all uuids have lower-cased letters upon ingestion into the dss). Characters other than letters, numbers, and dashes are not allowed and will error. The specified character(s) will return all available bundle uuids starting with that character(s).
- **token** (*typing.Union[str, NoneType]*) – Token to manage retries. End users constructing queries should not set this parameter.
- **per_page** (*typing.Union[str, NoneType]*) – Max number of results to return per page.
- **search_after** (*typing.Union[str, NoneType]*) – **Search-After-Context**. An internal state pointer parameter for use with pagination. This parameter is referenced by the `Link` header as described in the "Pagination" section. The API client should not need to set this parameter directly; it should instead directly fetch the URL given in the `Link` header.

Lists all the bundles available in the data-store, responses will be returned in a paginated format, at most 500 values shall be returned at a time. Tombstoned bundles will be omitted from the list of bundles available.

classmethod get_bundles_checkout(*client*, *replica: str = None*, *checkout_job_id: str = None*)

Check the status of a checkout request.

Parameters

- **replica** (*<class 'str'>*) – Replica to fetch from.
- **checkout_job_id** (*<class 'str'>*) – A RFC4122-compliant ID for the checkout job request.

Use this route with the `checkout_job_id` identifier returned by `POST /bundles/{uuid}/checkout`.

classmethod get_collection(*client*, *uuid: str = None*, *replica: str = None*, *version: Optional[str] = None*)

Retrieve a collection given a UUID.

Parameters

- **uuid** (*<class 'str'>*) – A RFC4122-compliant ID for the collection.
- **replica** (*<class 'str'>*) – Replica to fetch from.
- **version** (*typing.Union[str, NoneType]*) – Timestamp of collection creation in DSS_VERSION format. If this is not provided, the latest version is returned.

Given a collection UUID, return the associated collection object.

classmethod get_collections(*client*, *per_page: Optional[str] = 500*, *start_at: Optional[str] = None*)

Retrieve a user's collections.

Pagination

This method supports pagination. Use `DSSClient.get_collections.iterate(**kwargs)` to create a generator that yields all results, making multiple requests over the wire if necessary:

```
for result in DSSClient.get_collections.iterate(**kwargs):
    ...
```

The keyword arguments for `DSSClient.get_collections.iterate()` are identical to the arguments for `DSSClient.get_collections()` listed here.

Parameters

- **per_page** (*typing.Union[str, NoneType]*) – Max number of results to return per page.
- **start_at** (*typing.Union[str, NoneType]*) – An internal state pointer parameter for use with pagination. This parameter is referenced by the `Link` header as described in the "Pagination" section. The API client should not need to set this parameter directly; it should instead directly fetch the URL given in the `Link` header.

Return a list of a user's collections. Collections are sets of links to files, bundles, other collections, or fragments of JSON metadata files. Each entry in the input set of links is checked for referential integrity (the link target must exist in the replica referenced). Up to 1000 items can be referenced in a new collection, or added or removed using `PATCH /collections`. New collections are private to the authenticated user. Collection items are de-duplicated (if an identical item is given multiple times, it will only be added once). Collections are replicated across storage replicas similarly to files and bundles.

classmethod get_event(*client*, *uuid: str = None*, *version: str = None*, *replica: str = None*)

Retrieve a bundle metadata document given a UUID and version.

Parameters

- **uuid** (*<class 'str'>*) – Bundle unique ID.

- **version** (*<class 'str'>*) – Timestamp of bundle creation in DSS_VERSION format.
- **replica** (*<class 'str'>*) – Replica to fetch from.

Given a bundle UUID and version, return the bundle metadata document.

classmethod get_events(*client*, *from_date: Optional[str] = None*, *to_date: Optional[str] = None*, *replica: str = None*, *per_page: Optional[str] = 1*, *token: Optional[str] = None*)

Replay events

Pagination

This method supports pagination. Use `DSSClient.get_events.iterate(**kwargs)` to create a generator that yields all results, making multiple requests over the wire if necessary:

```
for result in DSSClient.get_events.iterate(**kwargs):
    ...
```

The keyword arguments for `DSSClient.get_events.iterate()` are identical to the arguments for `DSSClient.get_events()` listed here.

Parameters

- **from_date** (*typing.Union[str, NoneType]*) – Timestamp to begin replaying events, in DSS_VERSION format. If this is not provided, replay from the earliest event.
- **to_date** (*typing.Union[str, NoneType]*) – Timestamp to stop replaying events, in DSS_VERSION format. If this is not provided, replay to the latest event.
- **replica** (*<class 'str'>*) – Replica to fetch from.
- **per_page** (*typing.Union[str, NoneType]*) – Max number of results to return per page.
- **token** (*typing.Union[str, NoneType]*) – Token to manage retries. End users constructing queries should not set this parameter.

Return urls where event data is available, with manifest of contents.

classmethod get_file(*client*, *uuid: str = None*, *replica: str = None*, *version: Optional[str] = None*, *token: Optional[str] = None*, *directurl: Optional[str] = None*, *content_disposition: Optional[str] = None*)

Retrieve a file given a UUID and optionally a version.

Streaming

Use `DSSClient.get_file.stream(**kwargs)` to get a `requests.Response` object whose body has not been read yet. This allows streaming large file bodies:

```
fid = "7a8fbda7-d470-467a-904e-5c73413fab3e"
with DSSClient().get_file.stream(uuid=fid, replica="aws") as fh:
    while True:
        chunk = fh.raw.read(1024)
        ...
        if not chunk:
            break
```

The keyword arguments for `DSSClient.get_file.stream()` are identical to the arguments for `DSSClient.get_file()` listed here.

Parameters

- **uuid** (*<class 'str'>*) – A RFC4122-compliant ID for the file.
- **replica** (*<class 'str'>*) – Replica to fetch from.
- **version** (*typing.Union[str, NoneType]*) – Timestamp of file creation in DSS_VERSION format. If this is not provided, the latest version is returned.
- **token** (*typing.Union[str, NoneType]*) – Token to manage retries. End users constructing queries should not set this parameter.
- **directurl** (*typing.Union[str, NoneType]*) – When set to true, the response will contain API-specific URLs that are tied to the specified replica, for example `gs://bucket/object` or `s3://bucket/object` The use of presigned URLs is recommended for data access. Cloud native URLs are currently provided for a limited set of use cases and may not be provided in the future. If cloud native URLs are required, please contact the data store team regarding the credentials necessary to use them.
- **content_disposition** (*typing.Union[str, NoneType]*) – Optional and does not work when directurl=true (only works with the default presigned url response). If this parameter is provided, the response from fetching the returned presigned url will include the specified Content-Disposition header. This can be useful to indicate to a browser that a file should be downloaded rather than opened in a new tab, and can also supply the original filename in the response. Example: .. code:: content_disposition="attachment; filename=data.json"

Given a file UUID, return the latest version of that file. If the version is provided, that version of the file is returned instead. Headers will contain the data store metadata for the file. This endpoint returns a HTTP redirect to another HTTP endpoint with the file contents.

classmethod get_subscription(*client*, *uuid: str = None*, *replica: str = None*, *subscription_type: Optional[str] = 'jmespath'*)

Retrieve an event subscription given a UUID.

Parameters

- **uuid** (*<class 'str'>*) – A RFC4122-compliant ID for the subscription.
- **replica** (*<class 'str'>*) – Replica to fetch from.
- **subscription_type** (*typing.Union[str, NoneType]*) – type of subscriptions to fetch (elasticsearch or jmespath)

Given a subscription UUID, return the associated subscription.

classmethod get_subscriptions(*client*, *replica: str = None*, *subscription_type: Optional[str] = 'jmespath'*)

Retrieve a user's event subscriptions.

Parameters

- **replica** (*<class 'str'>*) – Replica to fetch from.
- **subscription_type** (*typing.Union[str, NoneType]*) – Type of subscriptions to fetch (elasticsearch or jmespath).

Return a list of associated subscriptions.

classmethod head_file(*client*, *uuid: str = None*, *replica: str = None*, *version: Optional[str] = None*)

Retrieve a file's metadata given an UUID and optionally a version.

Parameters

- **uuid** (*<class 'str'>*) – A RFC4122-compliant ID for the file.
- **replica** (*<class 'str'>*) – Replica to fetch from.
- **version** (*typing.Union[str, NoneType]*) – Timestamp of file creation in DSS_VERSION format. If this is not provided, the latest version is returned.

Given a file UUID, return the metadata for the latest version of that file. If the version is provided, that version's metadata is returned instead. The metadata is returned in the headers. NOTE When using the HCA CLI, this will stream the file to stdout and may need to be piped. For example, `hca dss get-file --uuid UUID --replica aws > result.txt`

static load_swagger_json(*swagger_json*, *ptr_str='$ref'*)

Load the Swagger JSON and resolve {"$ref": "#/..."} internal JSON Pointer references.

login(*access_token=''*, *remote=False*)

Configure and save {prog} authentication credentials.

This command may open a browser window to ask for your consent to use web service authentication credentials.

Use –remote if using the CLI in a remote environment

logout()

Clear {prog} authentication credentials previously configured with `{prog} login`.

classmethod patch_bundle(*client*, *add_files: Optional[List[T]] = None*, *remove_files: Optional[List[T]] = None*, *uuid: str = None*, *replica: str = None*, *version: str = None*)

Update a bundle.

Parameters

- **add_files** (*typing.Union[typing.List, NoneType]*) – List of new files to add to the bundle. File names must be unique.
- **remove_files** (*typing.Union[typing.List, NoneType]*) – List of files to remove from the bundle. Files must match exactly to be removed. Files not found in the bundle are ignored.
- **uuid** (*<class 'str'>*) – A RFC4122-compliant ID of the bundle to update.
- **replica** (*<class 'str'>*) – Replica to update the bundle on. Updates are propagated to other replicas.
- **version** (*<class 'str'>*) – Timestamp of the bundle to update in DSS_VERSION format format (required).

Add or remove files from a bundle. A specific version of the bundle to update must be provided, and a new version will be written. Bundle manifests exceeding 20,000 files will not be included in the Elasticsearch index document.

classmethod patch_collection(*client*, *add_contents: Optional[List[T]] = None*, *description: Optional[str] = None*, *details: Optional[Mapping[KT, VT_co]] = None*, *name: Optional[str] = None*, *remove_contents: Optional[List[T]] = None*, *uuid: str = None*, *replica: str = None*, *version: str = None*)

Update a collection.

Parameters

- **add_contents** (*typing.Union[typing.List, NoneType]*) – List of new items to add to the collection. Items are de-duplicated (if an identical item is already present in the collection or given multiple times, it will only be added once).
- **description** (*typing.Union[str, NoneType]*) – New description for the collection.
- **details** (*typing.Union[typing.Mapping, NoneType]*) – New details for the collection.
- **name** (*typing.Union[str, NoneType]*) – New name for the collection.
- **remove_contents** (*typing.Union[typing.List, NoneType]*) – List of items to remove from the collection. Items must match exactly to be removed. Items not found in the collection are ignored.
- **uuid** (*<class 'str'>*) – A RFC4122-compliant ID of the collection to update.
- **replica** (*<class 'str'>*) – Replica to update the collection on. Updates are propagated to other replicas.
- **version** (*<class 'str'>*) – Timestamp of the collection to update in DSS_VERSION format format (required).

Add or remove items from a collection. A specific version of the collection to update must be provided, and a new version will be written.

classmethod post_bundles_checkout(*client*, *destination: Optional[str] = None*, *email: Optional[str] = None*, *uuid: str = None*, *version: Optional[str] = None*, *replica: str = None*)

Check out a bundle to DSS-managed or user-managed cloud object storage destination

Parameters

- **destination** (*typing.Union[str, NoneType]*) – User-owned destination storage bucket.
- **email** (*typing.Union[str, NoneType]*) – An email address to send status updates to.
- **uuid** (*<class 'str'>*) – A RFC4122-compliant ID for the bundle.
- **version** (*typing.Union[str, NoneType]*) – Timestamp of file creation in DSS_VERSION format. If this is not provided, the latest version is returned.
- **replica** (*<class 'str'>*) – Replica to fetch from.

Initiate asynchronous checkout of a bundle. The response JSON contains a field, `checkout_job_id`, that can be used to query the status of the checkout via the `GET /bundles/checkout/{checkout_job_id}` API method. FIXME: document the error code returned when the bundle or specified version does not exist. TODO: After some time period, the data will be removed. TBD: This could be based on initial checkout time or last access time.

classmethod post_search(*client*, *es_query: Mapping[KT, VT_co] = None*, *output_format: Optional[str] = 'summary'*, *replica: str = None*, *per_page: Optional[str] = 100*, *search_after: Optional[str] = None*)

Find bundles by searching their metadata with an Elasticsearch query

---

Pagination

This method supports pagination. Use `DSSClient.post_search.iterate(**kwargs)` to create a generator that yields all results, making multiple requests over the wire if necessary:

```
for result in DSSClient.post_search.iterate(**kwargs):
    ...
```

The keyword arguments for `DSSClient.post_search.iterate()` are identical to the arguments for `DSSClient.post_search()` listed here.

Parameters

- **es_query** (*typing.Mapping*) – Elasticsearch query
- **output_format** (*typing.Union[str, NoneType]*) – Specifies the output format. The default format, `summary`, is a list of UUIDs for bundles that match the query. Set this parameter to `raw` to get the verbatim JSON metadata for bundles that match the query. When using `output_format raw` the `per_page` size is limit to no more than 10 to avoid excessively large response sizes.
- **replica** (*<class 'str'>*) – Replica to search.
- **per_page** (*typing.Union[str, NoneType]*) – Max number of results to return per page. When using `output_format raw` the `per_page` size is limit to no more than 10 to avoid excessively large response sizes.
- **search_after** (*typing.Union[str, NoneType]*) – **Search-After-Context**. An internal state pointer parameter for use with pagination. This parameter is referenced by the `Link` header as described in the "Pagination" section. The API client should not need to set this parameter directly; it should instead directly fetch the URL given in the `Link` header.

Accepts Elasticsearch JSON query and returns matching bundle identifiers

The metadata seach index is implemented as a document-oriented database using Elasticsearch. The index stores all information relevant to a bundle within each bundle document, largely eliminating the need for object-relational mapping. This design is optimized for queries that filter the data.

To illustrate this concept, say our index stored information on three entities, `foo`, `bar`, and `baz`. A foo can have many bars and bars can have many bazes. If we were to index bazes in a document-oriented design, the information on the foo a bar comes from and the bazes it contains are combined into a single document. A example sketch of this is shown below in JSON-schema.

```
{
  "definitions": {
    "bar": {
      "type": "object",
      "properties": {
        "uuid": {
          "type": "string",
          "format": "uuid"
        },
        "foo": {
          "type": "object",
          "properties": {
            "uuid": {
              "type": "string",
              "format": "uuid"
```

(continues on next page)

(continued from previous page)

```
            },
            ...
          }
        },
        "bazes": {
          "type": "array",
          "items": {
            "type": "string",
            "format": "uuid"
          }
        },
        ...
      }
    }
  }
}
```

This closely resembles the structure of DSS bundle documents: projects have many bundles and bundles have many files. Each bundle document is a concatenation of the metadata on the project it belongs to and the files it contains.

There are limitations to the design of DSS's metadata search index. A few important ones are listed below.

- Joins between bundle metadata must be conducted client-side
- Querying is schema-specific; fields or values changed between schema version will break queries that use those fields and values
- A new search index must be built for each schema version
- A lot of metadata is duplicated between documents

classmethod put_bundle(*client*, *creator_uid: int = None*, *files: List[T] = None*, *uuid: str = None*, *version: str = None*, *replica: str = None*)

Create a bundle

Parameters

- **creator_uid** (*<class 'int'>*) – User ID who is creating this bundle.
- **files** (*typing.List*) – This is a list of dictionaries describing each of the files. Each dictionary includes the fields: - The "uuid" of a file already previously uploaded with "PUT file/{uuid}". - The "version" timestamp of the file. - The "name" of the file. This can be most anything, and is the name the file will have when downloaded. - The "indexed" field, which specifies whether a file should be indexed or not. Bundle manifests exceeding 20,000 files will not be included in the Elasticsearch index document. Example representing 2 files with dummy values: [{'uuid': 'ce55fd51-7833-469b-be0b-5da88ebebfcd', 'version': '2017-06-16T193604.240704Z', 'name': 'dinosaur_dna.fa', 'indexed': False}, {'uuid': 'ae55fd51-7833-469b-be0b-5da88ebebfca', 'version': '0303-04-23T193604.240704Z', 'name': 'dragon_dna.fa', 'indexed': False}]
- **uuid** (*<class 'str'>*) – A RFC4122-compliant ID for the bundle.
- **version** (*<class 'str'>*) – Timestamp of bundle creation in DSS_VERSION format.
- **replica** (*<class 'str'>*) – Replica to write to.

Create a new version of a bundle with a given UUID. The list of file UUID and versions to be included must be provided.

classmethod put_collection(*client*, *contents: List[T] = None*, *description: str = None*, *details: Mapping[KT, VT_co] = None*, *name: str = None*, *replica: str = None*, *uuid: str = None*, *version: str = None*)

Create a collection.

Parameters

- **contents** (*typing.List*) – A list of objects describing links to files, bundles, other collections, and metadata fragments that are part of the collection.
- **description** (*<class 'str'>*) – A long description of the collection, formatted in Markdown.
- **details** (*typing.Mapping*) – Supplementary JSON metadata for the collection.
- **name** (*<class 'str'>*) – A short name identifying the collection.
- **replica** (*<class 'str'>*) – Replica to write to.
- **uuid** (*<class 'str'>*) – A RFC4122-compliant ID for the collection.
- **version** (*<class 'str'>*) – Timestamp of collection creation in DSS_VERSION format.

Create a new collection. Collections are sets of links to files, bundles, other collections, or fragments of JSON metadata files. Each entry in the input set of links is checked for referential integrity (the link target must exist in the replica referenced). Up to 1000 items can be referenced in a new collection, or added or removed using `PATCH /collections`. New collections are private to the authenticated user. Collection items are de-duplicated (if an identical item is given multiple times, it will only be added once). Collections are replicated across storage replicas similarly to files and bundles.

classmethod put_file(*client*, *creator_uid: int = None*, *source_url: str = None*, *uuid: str = None*, *version: str = None*)

Create a new version of a file

Parameters

- **creator_uid** (*<class 'int'>*) – User ID who is creating this file.
- **source_url** (*<class 'str'>*) – Cloud bucket URL for source data. Example is "s3://bucket_name/serious_dna.fa" .
- **uuid** (*<class 'str'>*) – A RFC4122-compliant ID for the file.
- **version** (*<class 'str'>*) – Timestamp of file creation in DSS_VERSION format. If this is not provided, the latest version is returned.

Create a new version of a file with a given UUID. The contents of the file are provided by the client by reference using a cloud object storage URL. The file on the cloud object storage service must have metadata set listing the file checksums and content-type. The metadata fields required are:

- hca-dss-sha256: SHA-256 checksum of the file
- hca-dss-sha1: SHA-1 checksum of the file
- hca-dss-s3_etag: S3 ETAG checksum of the file. See https://stackoverflow.com/q/12186993 for the general algorithm for how checksum is calculated. For files smaller than 64MB, this is the MD5 checksum of the file. For files larger than 64MB but smaller than 640,000MB, we use 64MB chunks. For files larger than 640,000MB, we use a chunk size equal to the total file size divided by 10000, rounded up to the nearest MB. MB, in this section, refers to 1,048,576 bytes. Note that 640,000MB is not the same as 640GB!
- hca-dss-crc32c: CRC-32C checksum of the file

classmethod put_subscription(*client*, *attachments: Optional[Mapping[KT, VT_co]] = None*, *callback_url: str = None*, *encoding: Optional[str] = 'application/json'*, *es_query: Optional[Mapping[KT, VT_co]] = None*, *form_fields: Optional[Mapping[KT, VT_co]] = {}*, *hmac_key_id: Optional[str] = None*, *hmac_secret_key: Optional[str] = None*, *jmespath_query: Optional[str] = None*, *method: Optional[str] = 'POST'*, *payload_form_field: Optional[str] = 'payload'*, *replica: str = None*)

Create an event subscription.

Parameters

- **attachments** (*typing.Union[typing.Mapping, NoneType]*) – The set of bundle metadata items to be included in the payload of a notification request to a subscription endpoint. Each property in this object represents an attachment to the notification payload. Each attachment will be a child property of the `attachments` property of the payload. The name of such a child property can be chosen freely provided it does not start with an underscore. For example, if the subscription is .. code:: { "attachments": { "taxon": { "type": "jmespath", "expression": "files.biomaterial_json.biomaterials[].content.biomaterial_core.ncbi_taxon_id[]" } } } the corresponding notification payload will contain the following entry .. code:: "attachments": { "taxon": [9606, 9606] } If a general error occurs during the processing of attachments, the notification will be sent with `attachments` containing only the reserved `_errors` attachment containing a string describing the error. If an error occurs during the processing of a specific attachment, the notification will be sent with all successfully processed attachments and additionally the `_errors` attachment containing an object with one property for each failed attachment. For example, .. code:: "attachments": { "taxon": [9606, 9606] "_errors" { "biomaterial": "Some error occurred" } } The value of the `attachments` property must be less than or equal to 128 KiB in size when serialized to JSON and encoded as UTF-8. If it is not, the notification will be sent with "attachments": { "_errors": "Attachments too large (131073 bytes)" }
- **callback_url** (*<class 'str'>*) – The subscriber's URL. An HTTP request is made to the specified URL for every attempt to deliver a notification to the subscriber. If the HTTP response code is 2XX, the delivery attempt is considered successful. Otherwise, more attempts will be made with an exponentially increasing delay between attempts, until an attempt is successful or the a maximum number of attempts is reached. Occasionally, duplicate notifications may be sent. It is up to the receiver of the notification to tolerate duplicate notifications.
- **encoding** (*typing.Union[str, NoneType]*) – The MIME type describing the encoding of the request body * `application/json` - the HTTP request body is the notification payload as JSON * `multipart/form-data` - the HTTP request body is a list of form fields, each consisting of a name and a corresponding value. See https://tools.ietf.org/html/rfc7578 for details on this encoding. The actual notification payload will be placed as JSON into a field of the name specified via `payload_form_field`.
- **es_query** (*typing.Union[typing.Mapping, NoneType]*) – An Elasticsearch query for restricting the set of bundles for which the subscriber is notified. The subscriber will only be notified for newly indexed bundles that match the given query. If this parameter is present the subscription will be of type `elasticsearch`, otherwise it will be of type `jmespath`.
- **form_fields** (*typing.Union[typing.Mapping, NoneType]*) – A collection of static form fields to be supplied in the request body, alongside the actual notification payload. The value of each field must be a string. For example, if the subscriptions has this property set to `{"foo" : "bar"}`, the corre-

sponding notification HTTP request body will consist of a multipart frame with two frames, .. code:: ——————-2769baffc4f24cbc83ced26aa0c2f712 Content-Disposition: form-data; name="foo" bar ——————-2769baffc4f24cbc83ced26aa0c2f712 Content-Disposition: form-data; name="payload" {"transaction_id": "301c9079-3b20-4311-a131-bcda9b7f08ba", "subscription_id": ... Since the type of this property is `object`, multi-valued fields are not supported. This property is ignored unless `encoding` is `multipart/form-data`.

- **hmac_key_id** (*typing.Union[str, NoneType]*) – An optional key ID to use with `hmac_secret_key`.
- **hmac_secret_key** (*typing.Union[str, NoneType]*) – The key for signing requests to the subscriber's URL. The signature will be constructed according to https://tools.ietf.org/html/draft-cavage-http-signatures and transmitted in the HTTP `Authorization` header.
- **jmespath_query** (*typing.Union[str, NoneType]*) – An JMESPath query for restricting the set of bundles for which the subscriber is notified. The subscriber will only be notified for new bundles that match the given query. If `es_query` is specified, the subscription will be of type `elasticsearch`. If `es_query` is not present, the subscription will be of type `jmespath`
- **method** (*typing.Union[str, NoneType]*) – The HTTP request method to use when delivering a notification to the subscriber.
- **payload_form_field** (*typing.Union[str, NoneType]*) – The name of the form field that will hold the notification payload when the request is made. If the default name of the payload field collides with that of a field in `form_fields`, this porperty can be used to rename the payload and avoid the collision. This property is ignored unless `encoding` is `multipart/form-data`.
- **replica** (*<class 'str'>*) – Replica to write to.

Register an HTTP endpoint that is to be notified when a given event occurs. Each user is allowed 100 subscriptions, a limit that may be increased in the future. Concerns about notification service limitations should be routed to the DSS development team.

upload(*src_dir*, *replica*, *staging_bucket*, *timeout_seconds=1200*, *no_progress=False*, *bundle_uuid=None*)

Upload a directory of files from the local filesystem and create a bundle containing the uploaded files.

Parameters

- **src_dir** (*str*) – file path to a directory of files to upload to the replica.
- **replica** (*str*) – the replica to upload to. The supported replicas are: *aws* for Amazon Web Services, and *gcp* for Google Cloud Platform. [aws, gcp]
- **staging_bucket** (*str*) – a client controlled AWS S3 storage bucket to upload from.
- **timeout_seconds** (*int*) – the time to wait for a file to upload to replica.
- **no_progress** (*bool*) – if set, will not report upload progress. Note that even if this flag is not set, progress will not be reported if the logging level is higher than INFO or if the session is not interactive.

Upload a directory of files from the local filesystem and create a bundle containing the uploaded files. This method requires the use of a client-controlled object storage bucket to stage the data for upload.

class hca.dss.**DSSFile**

Local representation of a file on the DSS

count()
Return number of occurrences of value.

classmethod for_bundle_manifest(*manifest_bytes*, *bundle_uuid*, *version*, *replica*)
Even though the bundle manifest is not a DSS file, we need to wrap its info in a DSSFile object for consistency and logging purposes.

index()
Return first index of value.

Raises ValueError if the value is not present.

indexed
Alias for field number 5

name
Alias for field number 0

replica
Alias for field number 6

sha256
Alias for field number 3

size
Alias for field number 4

uuid
Alias for field number 1

version
Alias for field number 2

class hca.dss.**TaskRunner**(*threads=2*)
A wrapper for ThreadPoolExecutor that tracks futures for you and allows dynamic submission of tasks.

submit(*info*, *task*, **args*, ***kwargs*)
Add task to be run.

Should only be called from the main thread or from tasks submitted by this method. :param info: Something printable :param task: A callable

wait_for_futures()
Wait for all submitted futures to finish.

Should only be called from the main thread.

6.2.2 Upload Service

6.2.3 Authorization and Authentication system

class hca.auth.**AuthClient**(*config=None*, *swagger_url=None*, ***session_kwargs*)
Human Cell Atlas Data Coordination Platform Fusillade app demo

Log in: Auth0

OIDC spec

The Fusillade API supports pagination in a manner consistent with the GitHub API, which is based on RFC 5988. When the results of an API call exceed the page size specified, the HTTP response will contain a `Link` header of the following form: `Link: <https://auth.data.humancellatlas.org/v1/users?per_page=30&next_token=123>; rel="next"`. The URL in the header refers to the next page of

the results to be fetched; if no `Link rel="next"` URL is included, then all results have been fetched. The client should recognize and parse the `Link` header appropriately according to RFC 5988, and retrieve the next page if requested by the user, or if all results are being retrieved.

clear_cache()
Clear the cached API definitions for a component. This can help resolve errors communicating with the API.

classmethod delete_v1_group(*client*, *group_id: str = None*)
Remove a group from the system

Parameters **group_id** (*<class 'str'>*) – The name of the group.

Remove all users, policies, and roles from the group, and delete the group.

classmethod delete_v1_role(*client*, *role_id: str = None*)
Remove a role from the system

Parameters **role_id** (*<class 'str'>*) – The name of the role.

Remove the role from all users and groups, and finally delete the role.

expired_token()
Return True if we have an active session containing an expired (or nearly expired) token.

classmethod get_echo(*client*)
echoes the response

Echoes the response back.

classmethod get_login(*client*, *redirect_uri: str = None*, *state: Optional[str] = None*)
Establish the users identity using the OIDC provider

Parameters

- **redirect_uri** (*<class 'str'>*) – Where to redirect to once login is complete.
- **state** (*typing.Union[str, NoneType]*) – An opaque parameter that is returned back to the `redirect_uri`.

Send the user agent to an identity provider selector and generate a user account to establish the user's identity. This is a redirect endpoint.

classmethod get_logout(*client*, *client_id: Optional[str] = None*)
Logout the user from current sessions with the OIDC provider.

Parameters **client_id** (*typing.Union[str, NoneType]*) –

Logout the user from current sessions with the OIDC provider. You can log the users out from a specific application if the you know the client_id for the application. Otherwise the user will be logged out of the default application by oauth2_config.

classmethod get_oauth_authorize(*client*, *redirect_uri: Optional[str] = None*, *state: Optional[str] = None*, *client_id: Optional[str] = None*, *scope: Optional[str] = None*, *respone_type: Optional[str] = None*, *nonce: Optional[str] = None*, *prompt: Optional[str] = None*)
See [Auth Request](https://openid.net/specs/openid-connect-core-1_0.html#AuthRequest)

Streaming

Use `AuthClient.get_oauth_authorize.stream(**kwargs)` to get a `requests.Response` object whose body has not been read yet. This allows streaming large file bodies:

```
fid = "7a8fbda7-d470-467a-904e-5c73413fab3e"
with DSSClient().get_file.stream(uuid=fid, replica="aws") as fh:
    while True:
        chunk = fh.raw.read(1024)
        ...
        if not chunk:
            break
```

The keyword arguments for `AuthClient.get_oauth_authorize.stream()` are identical to the arguments for `AuthClient.get_oauth_authorize()` listed here.

Parameters

- **redirect_uri** (*typing.Union[str, NoneType]*) –
- **state** (*typing.Union[str, NoneType]*) –
- **client_id** (*typing.Union[str, NoneType]*) –
- **scope** (*typing.Union[str, NoneType]*) –
- **respone_type** (*typing.Union[str, NoneType]*) –
- **nonce** (*typing.Union[str, NoneType]*) –
- **prompt** (*typing.Union[str, NoneType]*) –

This endpoint is part of OIDC and is used to redirect to an openid provider. See Auth Request

classmethod get_oauth_userinfo(*client*)
See [User Info](https://openid.net/specs/openid-connect-core-1_0.html#UserInfo)

This endpoint is part of OIDC and is used to redirect to an openid provider. See User Info

classmethod get_openid_configuration(*client*, *host: str = None*)
See documentation at [Provider Config](https://openid.net/specs/openid-connect-discovery-1_0.html#ProviderConfig)

Parameters **host** (*<class 'str'>*) – Must be `auth.data.humancellatlas.org`.

This endpoint is part of OIDC, see documentation at Provider Config

classmethod get_v1_group(*client*, *group_id: str = None*)
Get properties of a group

Parameters **group_id** (*<class 'str'>*) – The name of the group.

Get properties of a group, including the group's IAM policy.

classmethod get_v1_group_roles(*client*, *group_id: str = None*, *next_token: Optional[str] = None*, *per_page: Optional[str] = None*)
Retrieve role(s) for a group

Pagination

This method supports pagination. Use `AuthClient.get_v1_group_roles.iterate(**kwargs)` to create a generator that yields all results, making multiple requests over the wire if necessary:

```
for result in AuthClient.get_v1_group_roles.iterate(**kwargs):
    ...
```

The keyword arguments for `AuthClient.get_v1_group_roles.iterate()` are identical to the arguments for `AuthClient.get_v1_group_roles()` listed here.

Parameters

- **group_id** (*<class 'str'>*) – The name of the group.
- **next_token** (*typing.Union[str, NoneType]*) –
- **per_page** (*typing.Union[str, NoneType]*) –

Paginate through all roles assigned to the group.

classmethod get_v1_group_users(*client*, *group_id: str = None*, *next_token: Optional[str] = None*, *per_page: Optional[str] = None*)

Retrieve user(s) in a group

Pagination

This method supports pagination. Use `AuthClient.get_v1_group_users.iterate(**kwargs)` to create a generator that yields all results, making multiple requests over the wire if necessary:

```
for result in AuthClient.get_v1_group_users.iterate(**kwargs):
    ...
```

The keyword arguments for `AuthClient.get_v1_group_users.iterate()` are identical to the arguments for `AuthClient.get_v1_group_users()` listed here.

Parameters

- **group_id** (*<class 'str'>*) – The name of the group.
- **next_token** (*typing.Union[str, NoneType]*) –
- **per_page** (*typing.Union[str, NoneType]*) –

Paginate through all users in a group.

classmethod get_v1_groups(*client*, *next_token: Optional[str] = None*, *per_page: Optional[str] = None*)

List groups

Pagination

This method supports pagination. Use `AuthClient.get_v1_groups.iterate(**kwargs)` to create a generator that yields all results, making multiple requests over the wire if necessary:

```
for result in AuthClient.get_v1_groups.iterate(**kwargs):
    ...
```

The keyword arguments for `AuthClient.get_v1_groups.iterate()` are identical to the arguments for `AuthClient.get_v1_groups()` listed here.

Parameters

- **next_token** (*typing.Union[str, NoneType]*) –

- **per_page** (*typing.Union[str, NoneType]*) –

Paginate through all groups.

classmethod get_v1_role(*client*, *role_id: str = None*)
Get properties of a role

Parameters **role_id** (*<class 'str'>*) – The name of the role.

Get properties of a role.

classmethod get_v1_roles(*client*, *next_token: Optional[str] = None*, *per_page: Optional[str] = None*)
List roles

Pagination

This method supports pagination. Use `AuthClient.get_v1_roles.iterate(**kwargs)` to create a generator that yields all results, making multiple requests over the wire if necessary:

```
for result in AuthClient.get_v1_roles.iterate(**kwargs):
    ...
```

The keyword arguments for `AuthClient.get_v1_roles.iterate()` are identical to the arguments for `AuthClient.get_v1_roles()` listed here.

Parameters

- **next_token** (*typing.Union[str, NoneType]*) –
- **per_page** (*typing.Union[str, NoneType]*) –

Paginate through all roles.

classmethod get_v1_user(*client*, *user_id: str = None*)
Retrieve information about a user

Parameters **user_id** (*<class 'str'>*) – User ID (email).

Retrieve information about the user's status and the policies attached.

classmethod get_v1_user_groups(*client*, *user_id: str = None*, *next_token: Optional[str] = None*, *per_page: Optional[str] = None*)
Retrieve group(s) for a user

Pagination

This method supports pagination. Use `AuthClient.get_v1_user_groups.iterate(**kwargs)` to create a generator that yields all results, making multiple requests over the wire if necessary:

```
for result in AuthClient.get_v1_user_groups.iterate(**kwargs):
    ...
```

The keyword arguments for `AuthClient.get_v1_user_groups.iterate()` are identical to the arguments for `AuthClient.get_v1_user_groups()` listed here.

Parameters

- **user_id** (*<class 'str'>*) – User ID (email).
- **next_token** (*typing.Union[str, NoneType]*) –
- **per_page** (*typing.Union[str, NoneType]*) –

Paginate through a list of groups of which a user is a member.

classmethod get_v1_user_owns(*client*, *user_id: str = None*, *next_token: Optional[str] = None*, *per_page: Optional[str] = None*, *resource_type: str = None*)

Retrieve the resources owned by the user

Pagination

This method supports pagination. Use `AuthClient.get_v1_user_owns.iterate(**kwargs)` to create a generator that yields all results, making multiple requests over the wire if necessary:

```
for result in AuthClient.get_v1_user_owns.iterate(**kwargs):
    ...
```

The keyword arguments for `AuthClient.get_v1_user_owns.iterate()` are identical to the arguments for `AuthClient.get_v1_user_owns()` listed here.

Parameters

- **user_id** (*<class 'str'>*) – User ID (email).
- **next_token** (*typing.Union[str, NoneType]*) –
- **per_page** (*typing.Union[str, NoneType]*) –
- **resource_type** (*<class 'str'>*) –

Paginate through a list of resources owned by a user.

classmethod get_v1_user_roles(*client*, *user_id: str = None*, *next_token: Optional[str] = None*, *per_page: Optional[str] = None*)

Retrieve roles a user is in

Pagination

This method supports pagination. Use `AuthClient.get_v1_user_roles.iterate(**kwargs)` to create a generator that yields all results, making multiple requests over the wire if necessary:

```
for result in AuthClient.get_v1_user_roles.iterate(**kwargs):
    ...
```

The keyword arguments for `AuthClient.get_v1_user_roles.iterate()` are identical to the arguments for `AuthClient.get_v1_user_roles()` listed here.

Parameters

- **user_id** (*<class 'str'>*) – User ID (email).
- **next_token** (*typing.Union[str, NoneType]*) –
- **per_page** (*typing.Union[str, NoneType]*) –

Paginate through all roles attached to a user.

classmethod get_v1_users(*client*, *next_token: Optional[str] = None*, *per_page: Optional[str] = None*)

List users

Pagination

This method supports pagination. Use `AuthClient.get_v1_users.iterate(**kwargs)` to create a generator that yields all results, making multiple requests over the wire if necessary:

```
for result in AuthClient.get_v1_users.iterate(**kwargs):
    ...
```

The keyword arguments for `AuthClient.get_v1_users.iterate()` are identical to the arguments for `AuthClient.get_v1_users()` listed here.

Parameters

- **next_token** (*typing.Union[str, NoneType]*) –
- **per_page** (*typing.Union[str, NoneType]*) –

Paginate through all users.

static load_swagger_json(*swagger_json*, *ptr_str='$ref'*)

Load the Swagger JSON and resolve {"$ref": "#/..."} internal JSON Pointer references.

login(*access_token=''*, *remote=False*)

Configure and save {prog} authentication credentials.

This command may open a browser window to ask for your consent to use web service authentication credentials.

Use –remote if using the CLI in a remote environment

logout()

Clear {prog} authentication credentials previously configured with `{prog} login`.

classmethod post_oauth_revoke(*client*, *client_id: str = None*, *token: str = None*)

Revoke a refresh token

Parameters

- **client_id** (*<class 'str'>*) –
- **token** (*<class 'str'>*) – The refresh token to revoke.

Revokes a refresh token from a client making all future token refresh requests fail.

classmethod post_oauth_token(*client*)

Retrieve the authentications token

Streaming

Use `AuthClient.post_oauth_token.stream(**kwargs)` to get a `requests.Response` object whose body has not been read yet. This allows streaming large file bodies:

```
fid = "7a8fbda7-d470-467a-904e-5c73413fab3e"
with DSSClient().get_file.stream(uuid=fid, replica="aws") as fh:
    while True:
        chunk = fh.raw.read(1024)
        ...
        if not chunk:
            break
```

The keyword arguments for `AuthClient.post_oauth_token.stream()` are identical to the arguments for `AuthClient.post_oauth_token()` listed here.

This endpoint is part of OIDC and is used to redirect to an openid provider. See Token Endpoint, and Refresh Tokens

classmethod post_oauth_userinfo(*client*)
See [User Info](https://openid.net/specs/openid-connect-core-1_0.html#UserInfo)

This endpoint is part of OIDC and is used to redirect to an openid provider. See User Info

classmethod post_v1_group(*client*, *group_id: str = None*, *policy: Optional[Mapping[KT, VT_co]] = None*, *roles: Optional[List[T]] = None*)
Create a new group

Parameters

- **group_id** (*<class 'str'>*) – Used to identify users, groups, and roles.
- **policy** (*typing.Union[typing.Mapping, NoneType]*) –
- **roles** (*typing.Union[typing.List, NoneType]*) –

Create a new group, attach an IAM policy, and assign roles.

classmethod post_v1_policies_evaluate(*client*, *principal: str = None*, *action: List[T] = None*, *resource: List[T] = None*)
Evaluate a user's permissions

Parameters

- **principal** (*<class 'str'>*) – Attested user identifier.
- **action** (*typing.List*) – The action the principal is attempting to perform.
- **resource** (*typing.List*) – The resource the principal will perform the action against.

Given a set of principals, actions, and resources, return a set of access control decisions.

classmethod post_v1_role(*client*, *role_id: str = None*, *policy: Mapping[KT, VT_co] = None*)
Create a new role

Parameters

- **role_id** (*<class 'str'>*) – Used to identify users, groups, and roles.
- **policy** (*typing.Mapping*) –

Create a new role and attach a IAM policy.

classmethod post_v1_user(*client*, *user_id: str = None*, *groups: Optional[List[T]] = None*, *roles: Optional[List[T]] = None*, *policy: Optional[Mapping[KT, VT_co]] = None*)
Create a new user

Parameters

- **user_id** (*<class 'str'>*) – Used to identify users, groups, and roles.
- **groups** (*typing.Union[typing.List, NoneType]*) –
- **roles** (*typing.Union[typing.List, NoneType]*) –
- **policy** (*typing.Union[typing.Mapping, NoneType]*) –

Create a new user with the specified groups, roles, and iam policy.

classmethod put_v1_group_policy(*client*, *policy: Optional[Mapping[KT, VT_co]] = None*, *group_id: str = None*)

Modify policy

Parameters

- **policy** (*typing.Union[typing.Mapping, NoneType]*) –
- **group_id** (*<class 'str'>*) – The name of the group.

Modify or create a policy attached to a group.

classmethod put_v1_group_role(*client*, *roles: Optional[List[T]] = None*, *group_id: str = None*, *action: str = None*)

Add or remove a group from role(s)

Parameters

- **roles** (*typing.Union[typing.List, NoneType]*) –
- **group_id** (*<class 'str'>*) – The name of the group.
- **action** (*<class 'str'>*) –

Modify the role(s) assigned to a group.

classmethod put_v1_group_user(*client*, *users: Optional[List[T]] = None*, *group_id: str = None*, *action: str = None*)

Add or remove user(s) from a group

Parameters

- **users** (*typing.Union[typing.List, NoneType]*) –
- **group_id** (*<class 'str'>*) – The name of the group.
- **action** (*<class 'str'>*) –

Modify the user(s) assigned to a group.

classmethod put_v1_role_policy(*client*, *policy: Optional[Mapping[KT, VT_co]] = None*, *role_id: str = None*)

Modify policy

Parameters

- **policy** (*typing.Union[typing.Mapping, NoneType]*) –
- **role_id** (*<class 'str'>*) – The name of the role.

Modify the IAM policy attached to the role.

classmethod put_v1_user(*client*, *user_id: str = None*, *status: str = None*)

Modify user status

Parameters

- **user_id** (*<class 'str'>*) – User ID (email).
- **status** (*<class 'str'>*) –

Enable or disable a user. A disabled user will return false for all evaluations with that user as principal.

classmethod put_v1_user_group(*client*, *groups: Optional[List[T]] = None*, *user_id: str = None*, *action: str = None*)

Add or remove a user from group(s)

Parameters

- **groups** (*typing.Union[typing.List, NoneType]*) –
- **user_id** (*<class 'str'>*) – User ID (email).
- **action** (*<class 'str'>*) –

Modify group(s) in which a user is a member.

classmethod put_v1_user_policy(*client*, *policy: Optional[Mapping[KT, VT_co]] = None*, *user_id: str = None*)

Modify policy

Parameters

- **policy** (*typing.Union[typing.Mapping, NoneType]*) –
- **user_id** (*<class 'str'>*) – User ID (email).

Modify or add the user's IAM policy.

classmethod put_v1_user_role(*client*, *roles: Optional[List[T]] = None*, *user_id: str = None*, *action: str = None*)

Add or remove user from role(s)

Parameters

- **roles** (*typing.Union[typing.List, NoneType]*) –
- **user_id** (*<class 'str'>*) – User ID (email).
- **action** (*<class 'str'>*) –

Modify the role(s) attached to a user.

Links: genindex / modindex / search

6.3 HCA Tutorials

This page contains tutorials for using the HCA tools in this repo.

6.3.1 Python Open Endpoint Examples

The HCA API provides several ways for users of the Human Cell Atlas (HCA) to access and download data sets from the HCA. This page covers how to access HCA data using Python API bindings.

NOTE: The HCA CLI utility is compatible with Python 3.5+.

6.3.1.1 `create_version`

Returns a timestamp in `DSS_VERSION` format (e.g., `1985-04-12T232050.520000Z`), necessary for versioning bundles or files.

Note: A version is a timestamp in RFC3339 format that keeps track of the most recent iteration of a bundle or file. A bundle is a collection of many different data files, and both bundles and files have version numbers.

Example call to `create_version()`:

```
from hca.dss import DSSClient

dss = DSSClient()

dss.create_version()
```

6.3.1.2 download

Downloads a bundle to the local filesystem as a directory. By default, both data and metadata files are downloaded. The `no_data` or `no_metadata` flags can be set to True to download only the metadata or data, respectively (see example below).

Implementation detail: All files are downloaded to a local cache directory called `.hca` that is created in the directory where the download is initiated. The user should never need to interact directly with the `.hca` directory.

See note above regarding version numbering.

Example call to `download()`:

```
from hca.dss import DSSClient

dss = DSSClient()

dss.download(
    bundle_uuid="ffffaf55-f19c-40e3-aa81-a6c69d357265",
    version="2019-08-01T200147.836832Z",
    replica="aws",
    download_dir="download_test",
)
```

Example response:

```
{
    "bundle": {
        "creator_uid": 8008,
        "files": [
          {
            "content-type": "application/json; dcp-type=\"metadata/biomaterial\"",
            "crc32c": "5c084696",
            "indexed": true,
            "name": "cell_suspension_0.json",
            "s3_etag": "bd60da05055d1cd544855dd35cb12470",
            "sha1": "fdeb52d3caf0becce0575528c81bf0a06cb4a023",
            "sha256":
↪"e0ff1c402a4d6c659937f90d00d9820a2ebf0ebc920260a2a2bddf0961c30de5",
            "size": 847,
            "uuid": "134c0f04-76ae-405d-aea4-b72c08a53dd9",
            "version": "2019-07-09T230754.589000Z"
          },
          {
```

(continues on next page)

(continued from previous page)

```
            "content-type": "application/json; dcp-type=\"metadata/biomaterial\"",
            "crc32c": "39e6f9e1",
            "indexed": true,
            "name": "specimen_from_organism_0.json",
            "s3_etag": "f30917f841530d78e16223354049c8dc",
            "sha1": "98171c05647a3b771afb3bd61e65d0a25b0afe7f",
            "sha256":
↪"35406f0b8fa1ece3e3589151978aefef28f358afa163874b286eab837fcabfca",
            "size": 864,
            "uuid": "577a91d8-e579-41b6-9353-7e4e774c161a",
            "version": "2019-07-09T222811.151000Z"
          },

          ...

          {
            "content-type": "application/gzip; dcp-type=data",
            "crc32c": "38f31e58",
            "indexed": false,
            "name": "SRR6579532_2.fastq.gz",
            "s3_etag": "ac67e10df687471f5808be96499836c6",
            "sha1": "8743feb4d1ce82328127d10e2b1dfa35e5ae4b5a",
            "sha256":
↪"3d788e06b5ca4c8fc679b47c790b1e266f73d48818a1749743ec85f096d657ea",
            "size": 43810957,
            "uuid": "1330ef1a-7a21-40c6-84c5-5cec18204028",
            "version": "2019-08-03T150636.729022Z"
          }
        ],
        "uuid": "ffffaf55-f19c-40e3-aa81-a6c69d357265",
        "version": "2019-08-01T200147.836832Z"
    }
}
```

Example call to `download()`, specifying the flags needed to download the data or the metadata only:

```
from hca.dss import DSSClient

dss = DSSClient()

UUID = "ffffaf55-f19c-40e3-aa81-a6c69d357265"
VERSION = "ffffaf55-f19c-40e3-aa81-a6c69d357265"

# Download the metadata only
dss.download(
    bundle_uuid=UUID,
    version=VERSION,
    replica="aws",
    download_dir=".hca_metadata_only"
)

# Download the data only
dss.download(
    bundle_uuid=UUID,
    version=VERSION,
    replica="aws",
    download_dir=".hca_data_only"
```

(continues on next page)

(continued from previous page)

```
)
```

6.3.1.3 download_manifest

Downloads a list of files specified in a user-provided manifest file.

The manifest file should be in TSV (tab-separated variable) format, with one line in the manifest per file to download. The manifest should contain information about files (one file per line). The information that must be provided for a given bundle is available from the `get_bundle()` method.

The header row must define the columns:

- `bundle_uuid` - UUID of the requested bundle
- `bundle_version` - the version of the requested bundle
- `file_name` - the name of the file as specified in the bundle
- `file_uuid` - the UUID of the file in the DSS
- `file_sha256` - the SHA-256 hash of the file
- `file_size` - the size of the file

Example call to `download_manifest()`:

```
from hca.dss import DSSClient
import os
from hca.util import tsv
import json
import pprint
from get_bundle_api import fetch_bundle, save_bundle, BUNDLE_JSON

dss = DSSClient()

if not os.path.isfile(BUNDLE_JSON):
    bundle = fetch_bundle()
    save_bundle(bundle)

with open("manifest.tsv", "w", newline='') as manifest:
    writer = tsv.DictWriter(
        manifest,
        fieldnames=(
            "bundle_uuid",
            "bundle_version",
            "file_name",
            "file_uuid",
            "file_version",
            "file_sha256",
            "file_size",
        )
    )
    writer.writeheader()

    with open(BUNDLE_JSON, "w") as jsonfile:
        try:
            data = json.load(jsonfile)
            bundle_uuid, bundle_version = (
```

(continues on next page)

(continued from previous page)

```
                data["bundle"]["uuid"],
                data["bundle"]["version"],
            )
            pprint.pprint(data)
            for content in data["bundle"]["files"]:
                if content["name"].endswith(".json"):
                    writer.writerow(
                        dict(
                            bundle_uuid=bundle_uuid,
                            bundle_version=bundle_version,
                            file_name=content["name"],
                            file_uuid=content["uuid"],
                            file_version=content["version"],
                            file_sha256=content["sha256"],
                            file_size=content["size"],
                        )
                    )
        except ValueError as e:
            print("Not a JSON file: %s" % e)

dss.download_manifest(replica="aws", manifest="manifest.tsv")
```

Example manifest TSV file:

```
bundle_uuid                             bundle_version              file_name
↪      file_uuid                                file_version                file_
↪sha256                                                          file_size   file_path
002aeac5-4d74-462d-baea-88f5c620cb50    2019-08-01T200147.836900Z   cell_suspension_0.
↪json  c14b99ea-d8e2-4c84-9dc2-ce2245d8a743    2019-07-09T231935.003000Z
↪b43cebcca9cd5213699acce7356d226de07edef5c5604510a697159af1a12149    847         .
↪hca/v2/files_2_4/b4/3ceb/
↪b43cebcca9cd5213699acce7356d226de07edef5c5604510a697159af1a12149
```

6.3.1.4 `file_head`

Returns the metadata for the latest version of a file with a given UUID. If the version is provided, the metadata for that specific version is returned instead. The metadata is returned in the headers.

Example call to `file_head()`:

```
from hca.dss import DSSClient

dss = DSSClient()

print("Calling dss.head_file() with a file UUID:")
response = dss.head_file(
    uuid="6887bd52-8bea-47d9-bbd9-ff71e05faeee",
    replica="aws",
)
if response.status_code==200:
    print("Success!")
    print("Headers: %s"%(response.headers))

print()
```

(continues on next page)

(continued from previous page)

```
# Optionally, add a version
print("Calling dss.head_file() with a file UUID and version:")
response = dss.head_file(
    uuid="6887bd52-8bea-47d9-bbd9-ff71e05faeee",
    replica="aws",
    version="2019-01-30T165057.189000Z",
)
if response.status_code==200:
    print("Success!")
    print("Headers: %s"%(response.headers))
```

Example JSON header returned by API:

```
{
  "Date": "Tue, 22 Oct 2019 19:16:50 GMT",
  "Content-Type": "text/html; charset=utf-8",
  "Content-Length": "0",
  "Connection": "keep-alive",
  "x-amzn-RequestId": "bea3fd18-f373-4cb9-b0d2-0642c955eb5b",
  "X-DSS-SHA1": "ccac0f3fb16d1209ac88de8f293e61a115cfee38",
  "Access-Control-Allow-Origin": "*",
  "X-DSS-S3-ETAG": "d1634210a190ae78f6dd7a21f3c6ef1d",
  "X-DSS-SHA256": "24265fd0ebcdfe84eb1a09227c58c117ed03006b1de3f1e0694e50ed63b2f9e7",
  "Strict-Transport-Security": "max-age=31536000; includeSubDomains; preload",
  "Access-Control-Allow-Headers": "Authorization,Content-Type,X-Amz-Date,X-Amz-
↪Security-Token,X-Api-Key",
  "X-DSS-CONTENT-TYPE": 'application/json; dcp-type="metadata/biomaterial"',
  "X-DSS-CRC32C": "ec41da6a",
  "X-DSS-CREATOR-UID": "8008",
  "x-amz-apigw-id": "B-pROGlIoAMFUwg=",
  "X-DSS-VERSION": "2019-01-30T165057.189000Z",
  "X-Amzn-Trace-Id": "Root=1-5daf55a1-132caa16297ffc40a4046739;Sampled=0",
  "X-AWS-REQUEST-ID": "eeeb46a0-61a2-4fb5-aae9-21fe6a01f277",
  "X-DSS-SIZE": "856",
}
```

6.3.1.5 get_bundle

For a given bundle UUID and optionally a bundle version, returns information about the latest version of that bundle. Information returned includes the bundle creator, UUID, and version, as well as information about each file in the bundle, such as the file name, UUID, version, etc.

Example call to `get_bundle()`:

```
import os
import json
import subprocess
from hca.dss import DSSClient

BUNDLE_JSON = os.path.join(os.path.dirname(os.path.abspath(__file__)), 'data', 'get_
↪bundle.json')

def main():
    bundle = fetch_bundle()
    print_bundle(bundle)
```

(continues on next page)

(continued from previous page)

```
    save_bundle(bundle)

def fetch_bundle():
    dss = DSSClient()
    return dss.get_bundle(
        replica="aws", uuid="002aeac5-4d74-462d-baea-88f5c620cb50", version="2019-08-
↪01T200147.836900Z"
    )

def print_bundle(bundle):
    """Print a bundle and its contents to the console"""
    print("Bundle Contents:")
    for file in bundle["bundle"]["files"]:
        print(f"File: {json.dumps(file, indent=4)}")

    print(f'Bundle Creator: {bundle["bundle"]["creator_uid"]}')
    print(f'Bundle UUID   : {bundle["bundle"]["uuid"]}')
    print(f'Bundle Version: {bundle["bundle"]["version"]}')

def save_bundle(bundle):
    """Save bundle information to a JSON file. Useful for download_manifest_api.py␣
↪script."""
    if not os.path.exists("data"):
        subprocess.call(["mkdir", "data"])
    with open(BUNDLE_JSON, 'w') as f:
        f.write(json.dumps(bundle))

if __name__=="__main__":
    main()
```

Example of the JSON returned by `get_bundle()`:

```
{
  "bundle": {
    "creator_uid": 8008,
    "files": [
      {
        "name": "cell_suspension_0.json",
        "uuid": "c14b99ea-d8e2-4c84-9dc2-ce2245d8a743",
        "version": "2019-07-09T231935.003000Z"
        "content-type": "application/json; dcp-type=\"metadata/biomaterial\"",
        "crc32c": "892ad18b",
        "indexed": true,
        "s3_etag": "57814b3405165d975a6688dc8110dea0",
        "sha1": "849ebad4cff8f4fdf10ad25ad801ebb8aacc58b7",
        "sha256": "b43cebcca9cd5213699acce7356d226de07edef5c5604510a697159af1a12149",
        "size": 847,
      },
      {
        "name": "specimen_from_organism_0.json",
        "uuid": "05998af7-fa6f-44fe-bd16-ac8eafb42f28",
        "version": "2019-07-09T222953.739000Z"
        "content-type": "application/json; dcp-type=\"metadata/biomaterial\"",
        "crc32c": "8686eb38",
        "indexed": true,
        "s3_etag": "c3079914aa72f4aafa926594c756c978",
        "sha1": "885f0d6c524796116394fc4e60f0d9f65988765f",
```

(continues on next page)

(continued from previous page)

```
        "sha256": "d0c8cc0d13e30b73241405035d98265eab891ea94fbccc3da4bb0ca10c3d0f24",
        "size": 872,
      },
      ...
    ],
    "uuid": "002aeac5-4d74-462d-baea-88f5c620cb50",
    "version": "2019-08-01T200147.836900Z"
  }
}
```

6.3.1.6 get_bundles_checkout

Check the status and location of a checkout request.

Example call to `get_bundles_checkout()`:

```
from hca.dss import DSSClient

dss = DSSClient()

bundle_checkout_status = dss.get_bundles_checkout(
    replica="aws", checkout_job_id="4de1c603-fa8b-4c07-af37-06159e6951e0"
)

print(f'Bundle checkout status: {bundle_checkout_status["status"]}!')
if bundle_checkout_status["status"] == "SUCCEEDED":
    print(f'File is located at: {bundle_checkout_status["location"]}')
```

Example of the JSON returned by `get_bundles_checkout()`:

```
{
  "location": "s3://org-hca-dss-checkout-prod/bundles/fff54b87-26fe-42a9-be54-
↪3f5a7ef8176e.2019-03-26T131455.775610Z",
  "status": "SUCCEEDED"
}
```

6.3.1.7 get_file

Retrieves file metadata given a UUID, optionally a version. (To download a file, use the `hca download` command.)

Example call to `get_file()`:

```
from hca.dss import DSSClient
import json

dss = DSSClient()

json_response = dss.get_file(replica="aws", uuid="666ff3f0-67a1-4ead-82e9-3f96a8c0a9b1
↪")

for content in json_response:
    print(f"{content}: {json.dumps(json_response[content], indent=4)}")
```

Example of the JSON returned by `get_file()`:

```
{
  "describedBy": "https://schema.humancellatlas.org/type/file/7.0.2/sequence_file",
  "schema_type": "file",
  "file_core": {
    "file_name": "SRR6546754_2.fastq.gz",
    "file_format": "fastq.gz"
  },
  "read_index": "read2",
  "insdc_run": [
    "SRR6546754"
  ],
  "technical_replicate_group": "Rep_id_7031",
  "provenance": {
    "document_id": "39a93f75-0db3-4ee2-ab22-3eaa9932cf67",
    "submission_date": "2019-01-30T11:15:21.403Z",
    "update_date": "2019-02-19T17:17:10.540Z"
  }
}
```


6.3.1.8 `login`

Configures and saves authentication credentials.

Example call to `login()`:

```
from hca.dss import DSSClient

dss = DSSClient()

access_token = "test_access_token"
dss.login(access_token=access_token)
```

6.3.1.9 `logout`

Clears authentication credentials previously configured with login.

Example call to `logout()`:

```
from hca.dss import DSSClient

dss = DSSClient()

dss.logout()
```

6.3.1.10 `post_bundles_checkout`

Returns a `checkout-job-id` (e.g., `4de1c603-fa8b-4c07-af37-06159e6951e0`). This `checkout-job-id` can then be used with the `get_bundles_checkout()` method.

Example call to `post_bundles_checkout()`:

```
from hca.dss import DSSClient

dss = DSSClient()
```

(continues on next page)

(continued from previous page)

```
checkout_id = dss.post_bundles_checkout(uuid="fff746b3-e3eb-496a-88a3-5fa1fa358392",␣
↪replica="aws")
print(checkout_id)
```

6.3.1.11 post_search

Find bundles by their `bundle_fqid`, which is the bundle's UUID and version separated by a dot (.).

For example, the bundle FQID `fff807ba-bc98-4247-a560-49fb90c9675c.2019-08-01T200147.111027Z` is a bundle with the UUID `fff807ba-bc98-4247-a560-49fb90c9675c` and the version number `2019-08-01T200147.111027Z`.

This method returns an FQID and URL for each matching bundle.

Example call to `post_search()`:

```
from hca.dss import DSSClient

dss = DSSClient()

# Note:
# Passing es_query={} runs an empty search, which will match all bundles.

# Iterable post_search
for results in dss.post_search.iterate(replica="aws", es_query={}):
    print(results)
    break

# Non-iterable (first page only) post_search
print(dss.post_search(replica='aws', es_query={}))
```

Example output:

```
{
    ...
},
{
    "bundle_fqid": "fff807ba-bc98-4247-a560-49fb90c9675c.2019-08-01T200147.111027Z",
    "bundle_url": "https://dss.data.humancellatlas.org/v1/bundles/fff807ba-bc98-4247-
↪a560-49fb90c9675c?version=2019-08-01T200147.111027Z&replica=aws",
    "search_score": null
},
{
    ...
}
```

6.3.1.12 put_subscription, delete_subscription, get_subscription, get_subscriptions

- `get_subscriptions()`: Gets a list of user subscriptions.
- `put_subscription()`: Create a collection for the user given a replica and a call-back url.
- `get_subscription()`: Given the UUID of the subscription, show a subscription that the user created.

- `delete_subscription()`: Given a UUID and replica or the subscription, delete the subscription the user created.

Example API calls:

```
from hca import HCAConfig
from hca.dss import DSSClient

hca_config = HCAConfig()
hca_config["DSSClient"].swagger_url = f"https://dss.dev.data.humancellatlas.org/v1/
↪swagger.json"
dss = DSSClient(config=hca_config)

# Creates a sub based given a replica and a url
subscription = dss.put_subscription(
    replica="aws",
    callback_url=" https://dcp-cli-tutorials-put-delete-get-sub-api.humancellatlas.
↪org`"
)

callback, owner, replica, uuid = (
    subscription["callback_url"],
    subscription["owner"],
    subscription["replica"],
    subscription["uuid"],
)

# Lists all subs created
print(dss.get_subscriptions(replica="aws"))

# Lists a sub
print(dss.get_subscription(replica="aws", uuid=uuid))

# Deletes a sub based on a UUID
print(dss.delete_subscription(replica="aws", uuid=uuid))
```

6.3.1.13 `refresh_swagger`

Manually refresh the swagger document.

Links: genindex / modindex / search

6.3.2 CLI Open Endpoint Examples

The HCA CLI provides several ways for users of the Human Cell Atlas (HCA) to access and download data sets from the HCA. This page covers how to access the HCA using the `hca` command line utility.

NOTE: The HCA CLI utility is compatible with Python 3.5+.

6.3.2.1 `hca create-version`

Returns a timestamp in `DSS_VERSION` format (e.g., `1985-04-12T232050.520000Z`), necessary for versioning bundles or files.

Note: A version is a timestamp in RFC3339 format that keeps track of the most recent iteration of a bundle or file. A bundle is a collection of many different data files, and both bundles and files have version numbers.

Example call to `hca create-version`:

```
#!/usr/bin/env bash

hca dss create-version
```

6.3.2.2 hca download

Downloads a bundle to the local filesystem as a directory. By default, both data and metadata files are downloaded (flags can be added to download only the data or the metadata).

Implementation detail: All files are downloaded to a local cache directory called `.hca` that is created in the directory where the download is initiated. The user should never need to interact directly with the `.hca` directory.

See note above regarding version numbering.

Example call to `hca get-bundle`:

```
#!/usr/bin/env bash

hca dss download --replica aws --bundle-uuid ffffaf55-f19c-40e3-aa81-a6c69d357265 --
↪version 2019-08-01T200147.836832Z --download-dir download_test
```

Example response:

```
{
  "bundle": {
    "creator_uid": 8008,
    "files": [
      {
        "content-type": "application/json; dcp-type=\"metadata/biomaterial\"",
        "crc32c": "5c084696",
        "indexed": true,
        "name": "cell_suspension_0.json",
        "s3_etag": "bd60da05055d1cd544855dd35cb12470",
        "sha1": "fdeb52d3caf0becce0575528c81bf0a06cb4a023",
        "sha256": "e0ff1c402a4d6c659937f90d00d9820a2ebf0ebc920260a2a2bddf0961c30de5",
        "size": 847,
        "uuid": "134c0f04-76ae-405d-aea4-b72c08a53dd9",
        "version": "2019-07-09T230754.589000Z"
      },
      {
        "content-type": "application/json; dcp-type=\"metadata/biomaterial\"",
        "crc32c": "39e6f9e1",
        "indexed": true,
        "name": "specimen_from_organism_0.json",
        "s3_etag": "f30917f841530d78e16223354049c8dc",
        "sha1": "98171c05647a3b771afb3bd61e65d0a25b0afe7f",
        "sha256": "35406f0b8fa1ece3e3589151978aefef28f358afa163874b286eab837fcabfca",
        "size": 864,
        "uuid": "577a91d8-e579-41b6-9353-7e4e774c161a",
        "version": "2019-07-09T222811.151000Z"
```

(continues on next page)

(continued from previous page)

```
      },

      ...

      {
        "content-type": "application/gzip; dcp-type=data",
        "crc32c": "38f31e58",
        "indexed": false,
        "name": "SRR6579532_2.fastq.gz",
        "s3_etag": "ac67e10df687471f5808be96499836c6",
        "sha1": "8743feb4d1ce82328127d10e2b1dfa35e5ae4b5a",
        "sha256": "3d788e06b5ca4c8fc679b47c790b1e266f73d48818a1749743ec85f096d657ea",
        "size": 43810957,
        "uuid": "1330ef1a-7a21-40c6-84c5-5cec18204028",
        "version": "2019-08-03T150636.729022Z"
      }
    ],
    "uuid": "ffffaf55-f19c-40e3-aa81-a6c69d357265",
    "version": "2019-08-01T200147.836832Z"
  }
}
```

6.3.2.3 `hca download-manifest`

Downloads a list of files specified in a user-provided manifest file.

The manifest file should be in TSV (tab-separated variable) format, with one line in the manifest per file to download. The manifest should contain information about files (one file per line). The information that must be provided for a given bundle is available from the `get_bundle()` method.

The header row must define the columns:

- `bundle_uuid` - UUID of the requested bundle
- `bundle_version` - the version of the requested bundle
- `file_name` - the name of the file as specified in the bundle
- `file_uuid` - the UUID of the file in the DSS
- `file_sha256` - the SHA-256 hash of the file
- `file_size` - the size of the file

Example call to `hca download-manifest`:

```
#!/usr/bin/env bash

MANIFEST="manifest.tsv"

# Make the manifest file
cat /dev/null > ${MANIFEST}
echo -e "bundle_uuid\tbundle_version\tfile_name\tfile_uuid\tfile_version\tfile_
↪sha256\tfile_size\tfile_path\n" >> ${MANIFEST}
echo -e "ffffaf55-f19c-40e3-aa81-a6c69d357265\t2019-08-01T200147.836832Z\tlinks.
↪json\tdbf7bd27-b58e-431d-ba05-6a48f29e7cef\t2019-08-03T150636.
↪118831Z\tda4df14eb39cacdff01a08f27685534822c2d40adf534ea7b3e4adf261b9079a\t2081\t.
↪hca/v2/files_2_4/da/4df1/
↪da4df14eb39cacdff01a08f27685534822c2d40adf534ea7b3e4adf261b9079a\n" >> ${MANIFEST}
```

(continues on next page)

(continued from previous page)

```
echo "manifest.json file: ${MANIFEST}"

# Download files in the manifest
hca dss download-manifest --replica aws --manifest ${MANIFEST}
```

Example manifest TSV file:

```
bundle_uuid                                 bundle_version              file_name
↪      file_uuid                                    file_version                file_
↪sha256                                                                 file_size   file_path
002aeac5-4d74-462d-baea-88f5c620cb50    2019-08-01T200147.836900Z   cell_suspension_0.
↪json  c14b99ea-d8e2-4c84-9dc2-ce2245d8a743    2019-07-09T231935.003000Z
↪b43cebcca9cd5213699acce7356d226de07edef5c5604510a697159af1a12149    847         .
↪hca/v2/files_2_4/b4/3ceb/
↪b43cebcca9cd5213699acce7356d226de07edef5c5604510a697159af1a12149
```

6.3.2.4 `hca file-head`

Returns the metadata for the latest version of a file with a given UUID. If the version is provided, the metadata for that specific version is returned instead. The metadata is returned in the headers.

Example call to `hca file-head`:

```
#!/usr/bin/env bash

# Get the latest version
hca dss head-file --replica aws --uuid 666ff3f0-67a1-4ead-82e9-3f96a8c0a9b1

# Get the specified version
hca dss head-file --replica aws --uuid 6887bd52-8bea-47d9-bbd9-ff71e05faeee --version
↪2019-01-30T165057.189000Z
```

Example JSON header returned by API:

```
{
    "Date": "Tue, 22 Oct 2019 19:16:50 GMT",
    "Content-Type": "text/html; charset=utf-8",
    "Content-Length": "0",
    "Connection": "keep-alive",
    "x-amzn-RequestId": "bea3fd18-f373-4cb9-b0d2-0642c955eb5b",
    "X-DSS-SHA1": "ccac0f3fb16d1209ac88de8f293e61a115cfee38",
    "Access-Control-Allow-Origin": "*",
    "X-DSS-S3-ETAG": "d1634210a190ae78f6dd7a21f3c6ef1d",
    "X-DSS-SHA256": "24265fd0ebcdfe84eb1a09227c58c117ed03006b1de3f1e0694e50ed63b2f9e7
↪",
    "Strict-Transport-Security": "max-age=31536000; includeSubDomains; preload",
    "Access-Control-Allow-Headers": "Authorization,Content-Type,X-Amz-Date,X-Amz-
↪Security-Token,X-Api-Key",
    "X-DSS-CONTENT-TYPE": 'application/json; dcp-type="metadata/biomaterial"',
    "X-DSS-CRC32C": "ec41da6a",
    "X-DSS-CREATOR-UID": "8008",
    "x-amz-apigw-id": "B-pROGlIoAMFUwg=",
    "X-DSS-VERSION": "2019-01-30T165057.189000Z",
    "X-Amzn-Trace-Id": "Root=1-5daf55a1-132caa16297ffc40a4046739;Sampled=0",
```

(continues on next page)

(continued from previous page)

```
    "X-AWS-REQUEST-ID": "eeeb46a0-61a2-4fb5-aae9-21fe6a01f277",
    "X-DSS-SIZE": "856",
}
```

6.3.2.5 `hca get-bundle`

For a given bundle UUID and optionally a bundle version, returns information about the latest version of that bundle. Information returned includes the bundle creator, UUID, and version, as well as information about each file in the bundle, such as the file name, UUID, version, etc.

Example call to `hca get-bundle`:

```
#!/usr/bin/env bash

hca dss get-bundle --replica aws --uuid fff746b3-e3eb-496a-88a3-5fa1fa358392 --
↪version 2019-08-01T200147.130156Z
```

Example JSON returned by `hca get-bundle`:

```
{
  "bundle": {
    "creator_uid": 8008,
    "files": [
      {
        "name": "cell_suspension_0.json",
        "uuid": "c14b99ea-d8e2-4c84-9dc2-ce2245d8a743",
        "version": "2019-07-09T231935.003000Z"
        "content-type": "application/json; dcp-type=\"metadata/biomaterial\"",
        "crc32c": "892ad18b",
        "indexed": true,
        "s3_etag": "57814b3405165d975a6688dc8110dea0",
        "sha1": "849ebad4cff8f4fdf10ad25ad801ebb8aacc58b7",
        "sha256": "b43cebcca9cd5213699acce7356d226de07edef5c5604510a697159af1a12149",
        "size": 847,
      },
      {
        "name": "specimen_from_organism_0.json",
        "uuid": "05998af7-fa6f-44fe-bd16-ac8eafb42f28",
        "version": "2019-07-09T222953.739000Z"
        "content-type": "application/json; dcp-type=\"metadata/biomaterial\"",
        "crc32c": "8686eb38",
        "indexed": true,
        "s3_etag": "c3079914aa72f4aafa926594c756c978",
        "sha1": "885f0d6c524796116394fc4e60f0d9f65988765f",
        "sha256": "d0c8cc0d13e30b73241405035d98265eab891ea94fbccc3da4bb0ca10c3d0f24",
        "size": 872,
      },
      ...
    ],
    "uuid": "002aeac5-4d74-462d-baea-88f5c620cb50",
    "version": "2019-08-01T200147.836900Z"
  }
}
```

6.3.2.6 hca get-bundles-checkout

Check the status and location of a checkout request.

Example call to `hca get-bundles-checkout`:

```
#!/usr/bin/env bash

hca dss get-bundles-checkout --replica aws --checkout-job-id 4de1c603-fa8b-4c07-af37-
↪06159e6951e0
```

Example JSON returned by `hca get-bundles-checkout`:

```
{
  "location": "s3://org-hca-dss-checkout-prod/bundles/fff54b87-26fe-42a9-be54-
↪3f5a7ef8176e.2019-03-26T131455.775610Z",
  "status": "SUCCEEDED"
}
```

6.3.2.7 hca get-file

Retrieves a file given a UUID, optionally a version, and displays the details of the file.

Example call to `hca get-file`:

```
#!/usr/bin/env bash

hca dss get-file --replica aws --uuid 666ff3f0-67a1-4ead-82e9-3f96a8c0a9b1
```

Example JSON returned by `hca get-file`:

```
{
  "describedBy": "https://schema.humancellatlas.org/type/file/7.0.2/sequence_file",
  "schema_type": "file",
  "file_core": {
    "file_name": "SRR6546754_2.fastq.gz",
    "file_format": "fastq.gz"
  },
  "read_index": "read2",
  "insdc_run": [
    "SRR6546754"
  ],
  "technical_replicate_group": "Rep_id_7031",
  "provenance": {
    "document_id": "39a93f75-0db3-4ee2-ab22-3eaa9932cf67",
    "submission_date": "2019-01-30T11:15:21.403Z",
    "update_date": "2019-02-19T17:17:10.540Z"
  }
}
```

6.3.2.8 hca login

Configures and saves authentication credentials.

Example call to `hca login`:

```
#!/usr/bin/env bash

hca dss login --access-token test
```

6.3.2.9 hca logout

Clears authentication credentials previously configured with login.

Example call to `hca logout`:

```
#!/usr/bin/env bash

hca dss logout
```

6.3.2.10 hca post-bundles-checkout

Returns a `checkout-job-id` (e.g., `4de1c603-fa8b-4c07-af37-06159e6951e0`). This `checkout-job-id` can then be used with the `get_bundles_checkout()` method.

Example call to `hca post-bundles-checkout`:

```
#!/usr/bin/env bash

hca dss post-bundles-checkout --replica aws --uuid fff746b3-e3eb-496a-88a3-
↪5fa1fa358392
```

6.3.2.11 hca post-search

Find bundles by their `bundle_fqid`, which is the bundle's UUID and version separated by a dot (.).

For example, the bundle FQID `fff807ba-bc98-4247-a560-49fb90c9675c.2019-08-01T200147.111027Z` is a bundle with the UUID `fff807ba-bc98-4247-a560-49fb90c9675c` and the version number `2019-08-01T200147.111027Z`.

This method returns an FQID and URL for each matching bundle.

Example call to `hca post-search`:

```
#!/usr/bin/env bash

hca dss post-search --replica aws --es-query {} --no-paginate
```

Example output:

```
{
  ...
},
{
  "bundle_fqid": "fff807ba-bc98-4247-a560-49fb90c9675c.2019-08-01T200147.111027Z",
  "bundle_url": "https://dss.data.humancellatlas.org/v1/bundles/fff807ba-bc98-4247-
↪a560-49fb90c9675c?version=2019-08-01T200147.111027Z&replica=aws",
  "search_score": null
},
{
```

(continues on next page)

(continued from previous page)

```
  ...
}
```

6.3.2.12 `hca get-subscription(s), hca put-subscription, hca delete-subscription`

- `get_subscriptions()`: Gets a list of users subscription.
- `put_subscription()`: Create a collection for the user given a replica and a call-back url.
- `get_subscription()`: Given the UUID of the subscription, show a subscription that the user created.
- `delete_subscription()`: Given a UUID and rpelica or the subscription, delete the subscription the user created.

Example CLI calls:

```
#!/usr/bin/env bash

# Creates a sub based given a replica and a url
instance_info=$(hca dss put-subscription --callback-url https://dcp-cli-tutorials-put-
↪get-delete-sub.data.humancellatlas.org --replica aws)

ID=`echo ${instance_info} | jq -r '.uuid'`

echo $ID

# Lists all of subs created
hca dss get-subscriptions --replica aws

# List a sub
hca dss get-subscription --replica aws --uuid $ID

# Deletes a sub based on a UUID
hca dss delete-subscription --replica aws --uuid $ID
```

6.3.2.13 `hca refresh-swagger`

Manually refresh the swagger document.

```
#!/usr/bin/env bash

hca dss refresh-swagger
```

Links: genindex / modindex / search

6.3.3 Python Restricted Endpoint Examples

The HCA API provides several ways for users of the Human Cell Atlas (HCA) to access and download data sets from the HCA. This page covers how to access the HCA using Python API bindings.

The API calls listed here are restricted to those with upload or ingest permissions. Data will be submitted through a single Ingestion Service API. Submitted data will go through basic quality assurance before it is deposited into the Data Storage System (DSS) component.

In the document that follows, *privileged user* refers to a user with proper credentials and permission to upload/ingest data into the DSS.

NOTE: The HCA CLI utility is compatible with Python 3.5+.

6.3.3.1 delete_bundle

Deletes an existing bundle given a UUID, version, and replica.

Inputs:

- `uuid` - a unique, user-created UUID.
- `creator-uid` - a unique user ID (uid) for the bundle creator uid. This accepts integer values.
- `version` - a unique, user-created version number. Use the `create_verson()` API function to generate a `DSS_VERSION`.
- `replica` - which replica to use (corresponds to cloud providers; choices: `aws` or `gcp`)
- **`files` - a valid list of file objects, separated by commas (e.g., `[{<first_file>}, {<second_file>}, ... ]`). Ea**
  - Valid UUID of the file
  - Valid version number of the file
  - Name of the file
  - Boolean value - is this file indexed

Example call to `delete_bundle()`:

```
from hca import HCAConfig
from hca.dss import DSSClient

hca_config = HCAConfig()
hca_config["DSSClient"].swagger_url = f"https://dss.dev.data.humancellatlas.org/v1/
↪swagger.json"
dss = DSSClient(config=hca_config)

print(dss.delete_bundle(reason='test', uuid='98f6c379-cb78-4a61-9310-f8cc0341c0ea',␣
↪version='2019-08-02T202456.025543Z', replica='aws'))
```

6.3.3.2 put_bundle

Creates a bundle. A bundle can contain multiple files of arbitrary type.

Inputs:

- `uuid` - a unique, user-created UUID.
- `creator-uid` - a unique user ID (uid) for the bundle creator uid. This accepts integer values.
- `version` - a unique, user-created version number. Use the `create_verson()` API function to generate a `DSS_VERSION`.
- `replica` - which replica to use (corresponds to cloud providers; choices: `aws` or `gcp`)
- **`files` - a valid list of file objects, separated by commas (e.g., `[{<first_file>}, {<second_file>}, ... ]`). Ea**
  - Valid UUID of the file

- Valid version number of the file
- Name of the file
- Boolean value - is this file indexed

Example call to `put_bundle()`:

```
from hca import HCAConfig
from hca.dss import DSSClient
import os

hca_config = HCAConfig()

hca_config["DSSClient"].swagger_url = f"https://dss.dev.data.humancellatlas.org/v1/
↪swagger.json"
dss = DSSClient(config=hca_config)

dss.put_bundle(
    creator_uid=0,
    uuid="98f6c379-cb78-4a61-9310-f8cc0341c0ea",
    version="2019-08-02T202456.025543Z",
    replica="aws",
    files=[
        {
            "uuid": "2196a626-38da-4489-8b2f-645d342f6aab",
            "version": "2019-07-10T001103.121000Z",
            "name": "process_1.json1",
            "indexed": False,
        }
    ],
)
```

6.3.3.3 `patch_bundle`

Allows a user to modify an existing bundle. User passes in an optional list of files to add or remove from an existing bundle.

`add_files`/`remove_files` follow this format:

```
[
  {
    "path": "string",
    "type": "string",
    "uuid": "string",
    "version": "string"
  }
]
```

Example call to `patch_bundle()`:

```
from hca import HCAConfig
from hca.dss import DSSClient

hca_config = HCAConfig()
hca_config["DSSClient"].swagger_url = f"https://dss.dev.data.humancellatlas.org/v1/
↪swagger.json"
dss = DSSClient(config=hca_config)
```

(continues on next page)

(continued from previous page)

```
print(dss.patch_bundle(uuid='98f6c379-cb78-4a61-9310-f8cc0341c0ea', version='2019-08-
↪02T202456.025543Z', replica='aws'))
```

6.3.3.4 put_file

Creates a new version of a file, given an existing UUID, version, creator uid, and source URL.

Example call to `put_file()`:

```
from hca import HCAConfig
from hca.dss import DSSClient

hca_config = HCAConfig()

hca_config["DSSClient"].swagger_url = f"https://dss.dev.data.humancellatlas.org/v1/
↪swagger.json"
dss = DSSClient(config=hca_config)

print(
    dss.put_file(
        uuid="ead6434d-efb5-4554-98bc-027e160547c5",
        version="2019-07-30T174916.268875Z",
        creator_uid=0,
        source_url="s3://jeffwu-test/ead6434d-efb5-4554-98bc-027e160547c5/get_bundle.
↪json",
    )
)
```

6.3.3.5 put_collection, delete_collection, patch_collection, get_collection(s)

- `get_collection()` - Given a collection UUID, get the collection.
- `get_collections()` - Get a list of collections for a given user.
- `delete_collection()` - Given a collection UUID and replica, delete the collection from the replica.
- `put_collection()` - Create a collection.
- `patch_collection()` - Add or remove a given list of files from an existing collection.

To add or remove files with the API endpoints above, specify each file in the following format:

```
[
  {
    "path": "string",
    "type": "string",
    "uuid": "string",
    "version": "string"
  }
]
```

Example API calls:

```
from hca import HCAConfig
from hca.dss import DSSClient
import uuid
import os

hca_config = HCAConfig()
hca_config["DSSClient"].swagger_url = f"https://dss.dev.data.humancellatlas.org/v1/
↪swagger.json"
dss = DSSClient(config=hca_config)

# Creates a new collection
collection = dss.put_collection(
    uuid=str(uuid.uuid4()),
    version="2018-09-17T161441.564206Z",  # arbitrary
    description="foo",
    details={},
    replica="aws",
    name="bar",
    contents=[
        {
            "type": "bundle",
            "uuid": "ff818282-9735-45fa-a094-e9f2d3d0a954",  # overwrite if necessary
            "version": "2019-08-06T170839.843085Z",  # arbitrary
            "path": "https://dss.dev.data.humancellatlas.org/v1/bundles/ff818282-9735-
↪45fa-a094-e9f2d3d0a954?version=2019-08-06T170839.843085Z&replica=aws",
        }
    ],
)

uuid, version = collection["uuid"], collection["version"]

# Gets a list of collections
print(dss.get_collections(replica="aws"))

# Can add/remove files from a collection
print(dss.patch_collection(replica="aws", uuid=uuid, version=version))

# Gets a collection based on replcia and uuid
print(dss.get_collection(replica="aws", uuid=uuid))

# Deletes a colelction based on replica and uuid
print(dss.delete_collection(replica="aws", uuid=uuid))
```

6.3.3.6 upload

Uploads a directory of files from the local filesystem and creates a bundle containing the uploaded files.

Example call to `upload()`:

```
from hca import HCAConfig
from hca.dss import DSSClient
import boto3

s3 = boto3.resource('s3')
bucket = s3.Bucket('upload-test-unittest')
```

(continues on next page)

(continued from previous page)

```
hca_config = HCAConfig()
hca_config["DSSClient"].swagger_url = f"https://dss.dev.data.humancellatlas.org/v1/
↪swagger.json"
dss = DSSClient(config=hca_config)

print(dss.upload(src_dir="data/", replica="aws", staging_bucket="upload-test-unittest
↪"))

bucket.objects.all().delete()

print("Upload successful")
```

Links: genindex / modindex / search

6.3.4 CLI Restricted Endpoint Examples

The HCA CLI provides users of the Human Cell Atlas (HCA) to access and download data sets from the HCA. This page covers how to access the HCA using the HCA command line utility.

The CLI calls listed here are restricted to those with upload or ingest permissions. Data will be submitted through a single Ingestion Service API. Submitted data will go through basic quality assurance before it is deposited into the Data Storage System (DSS) component.

In the document that follows, *privileged user* refers to a user with proper credentials and permission to upload/ingest data into the DSS.

NOTE: The HCA CLI utility is compatible with Python 3.5+.

6.3.4.1 `hca delete-bundle`

Deletes an existing bundle given a UUID, version, and replica.

Example call to `hca delete-bundle`:

```
#!/usr/bin/env bash

hca dss delete-bundle --reason test --replica gcp --uuid 98f6c379-cb78-4a61-9310-
↪f8cc0341c0ea --version 2019-08-02T202456.025543Z
```

6.3.4.2 `hca put-bundle`

Creates a bundle. A bundle can contain multiple files of arbitrary type.

Inputs:

- `uuid` - a unique, user-created UUID.
- `creator-uid` - a unique user ID (uid) for the bundle creator uid. This accepts integer values.
- `version` - a unique, user-created version number. Use the `create_verson()` API function to generate a `DSS_VERSION`.
- `replica` - which replica to use (corresponds to cloud providers; choices: `aws` or `gcp`)
- **`files` - a valid list of file objects, separated by commas (e.g., `[{<first_file>}, {<second_file>}, ... ]`). Ea**

- Valid UUID of the file
- Valid version number of the file
- Name of the file
- Boolean value - is this file indexed

Example call to `put_bundle()`:

```
#!/usr/bin/env bash

hca dss put-bundle --creator-uid 0 --replica aws --uuid 98f6c379-cb78-4a61-9310-
↪f8cc0341c0ea --version 2019-08-02T202456.025543Z --files '{"uuid": "2196a626-38da-
↪4489-8b2f-645d342f6aab", "version": "2019-07-10T001103.121000Z" , "name": "process_
↪1.json", "indexed":false}'
```

6.3.4.3 hca patch-bundle

Allows user to pass in an optional list of files to add or remove from an exisiting bundle.

`add_files`/`remove_files` follow this format:

```
[
  {
    "path": "string",
    "type": "string",
    "uuid": "string",
    "version": "string"
  }
]
```

Example call to `hca patch-bundle`:

```
 #!/usr/bin/env bash

hca dss patch-bundle --replica aws --uuid 98f6c379-cb78-4a61-9310-f8cc0341c0ea --
↪version 2019-08-02T202456.025543Z
```

6.3.4.4 hca put-file

Creates a new version of a file, given an existing UUID, version, creator uid, and source URL.

Example call to `hca put-file`:

```
#!/usr/bin/env bash

hca dss put-file --uuid 38f6c379-cb78-4a61-9310-f8cc0341c0eb --version 2019-07-
↪30T164352.961501Z --creator-uid 0 --source-url s3://org-humancellatlas-dss-cli-test/
↪930a927d-0138-4a79-8c87-e45936fe4fc3/get_bundle.json
```

6.3.4.5 hca get-collection(s), hca put-collection, hca patch-collection, hca delete-collection

- `hca get-collection` - Given a collection UUID, get the collection.
- `hca get-collections` - Get a list of collections for a given user.

- `hca delete-collection` - Given a collection UUID and replica, delete the collection from the replica.
- `hca put-collection` - Create a collection.
- `hca patch-collection` - Add or remove a given list of files from an existing collection.

To add or remove files with the CLI actions above, specify each file in the following format:

```
[
  {
    "path": "string",
    "type": "string",
    "uuid": "string",
    "version": "string"
  }
]
```

Example CLI calls:

```
#!/usr/bin/env bash

info_instance=$(hca dss put-collection --uuid fff01947-bf94-43e9-86ca-f6ff6ae45d2c --
↪description foo --details {} --version 2018-09-17T161441.564206Z --replica aws --
↪name bar --contents '{"path": "https://dss.dev.data.humancellatlas.org/v1/bundles/
↪ff818282-9735-45fa-a094-e9f2d3d0a954?version=2019-08-06T170839.843085Z&replica=aws",
↪ "version": "2019-08-06T170839.843085Z", "type": "bundle", "uuid": "ff818282-9735-
↪45fa-a094-e9f2d3d0a954"}')

ID=`echo ${info_instance} | jq -r '.uuid'`
VERSION=`echo ${info_instance} | jq -r '.version'`

hca dss get-collections

hca dss patch-collection --replica aws --uuid $ID --verison $VERSION

hca dss get-collection --replica aws --uuid $ID

hca dss delete-collection --replica aws --uuid $ID
```

6.3.4.6 `hca upload`

Uploads a directory of files from the local filesystem and creates a bundle containing the uploaded files.

Example call to `hca upload`:

```
#!/usr/bin/env bash

hca dss upload --src-dir data/ --replica aws --staging-bucket upload-test-unittest

aws s3 rm s3://upload-test-unittest --recursive
```

Links: genindex / modindex / search

Links: genindex / modindex / search

Links: genindex / modindex / search

Python Module Index

h

Index