hca-cli Documentation

Release 0.1.0

James Mackey, Andrey Kislyuk

Aug 08, 2018

Contents

This repository contains a command line interface (CLI) and Python library for interacting with the Data Coordination Platform (DCP) of the Human Cell Atlas (HCA). Currently it allows interaction with the Upload Service and Data Storage Service (DSS).

CHAPTER 1

Installation

```
pip install hca.
```

CHAPTER 2

Usage

The hca package installs a command-line utility `hca`.

To see the list of commands you can use, type `hca --help`. Commands are grouped into major categories that roughly correspond to DCP system components, e.g. DSS, Staging Service. To get detailed help for a particular command group type, e.g. `hca upload --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.

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"
```

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
```

To use the Python interface with a local or test DSS, configure an HCAConfig object with a `swagger_url` field, and pass it as a parameter to the API client's constructor:

```
config = HCAConfig()
config['DSSClient'].swagger_url = "https://dss.example.com/v1/swagger.json"
client = DSSClient(config=config)
res = client.post_search(...)
```

CHAPTER 4

Testing

Before you run tests, first run `hca dss login`. This will pop up a browser and get you to authenticate with Google. Use an email from one of the whitelisted domains (in `DSS_SUBSCRIPTION_AUTHORIZED_DOMAINS_ARRAY` from here).

Then `make test`.

4.1 Bugs

Please report bugs, issues, feature requests, etc. on GitHub.

CHAPTER 5

License

Licensed under the terms of the MIT License.

CHAPTER 6

API documentation

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

Client for the Data Storage Service API.

classmethod delete_bundle()

Delete a bundle or a specific bundle version

Parameters

- **replica** (*<type 'str'>*) – Replica to write to.
- **uuid** (*<type 'str'>*) – A RFC4122-compliant ID for the bundle.
- **version** (*typing.Union[str, NoneType]*) – Timestamp of bundle creation in RFC3339.
- **reason** (*<type 'str'>*) – User-friendly reason for the bundle or timestamp-specfic bundle deletion.

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

classmethod delete_collection()

Delete a collection.

Parameters

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

Delete a collection.

classmethod delete_subscription()

Delete an event subscription.

Parameters

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

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=u''*, *dest_name=u''*, *metadata_files=(u'*',)*, *data_files=(u'*',)*, *initial_retries_left=10*, *min_delay_seconds=0.25*)

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

metadata_files (*–metadata-files* on the CLI) are 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_files (*–data-files* on the CLI) are 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. If and only if a data file matches any of the patterns in *data_files* will it be downloaded.

By default, all data and metadata files are downloaded. To disable the downloading of data files, use *–data-files ''* if using the CLI (or *data_files=()* if invoking *download* programmatically). Likewise for metadata files.

download_manifest(*manifest*, *replica*, *initial_retries_left=10*, *min_delay_seconds=0.25*)

Process the given manifest file in TSV (tab-separated values) format and download the files referenced by it.

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

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.

classmethod get_bundle()

Retrieve a bundle given a UUID and optionally a version.

Parameters

- **presignedurls** (*typing.Union[str, NoneType]*) – Include presigned URLs in the response. This is mutually exclusive with the directurls parameter.
- **uuid** (*<type 'str'>*) – Bundle unique ID.
- **directurls** (*typing.Union[str, NoneType]*) – Include direct-access URLs in the response. This is mutually exclusive with the presignedurls parameter.
- **token** (*typing.Union[str, NoneType]*) – Token to manage retries. End users constructing queries should not set this parameter.
- **version** (*typing.Union[str, NoneType]*) – Timestamp of bundle creation in RFC3339.
- **replica** (*<type 'str'>*) – Replica to fetch from.

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_checkout()

Check the status of a checkout request.

Parameters

- **replica** (*<type 'str'>*) – Replica to fetch from.
- **checkout_job_id** (*<type '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()
Retrieve a collection given a UUID.

Parameters

- **replica** (*<type 'str'>*) – Replica to fetch from.
- **uuid** (*<type 'str'>*) – A RFC4122-compliant ID for the collection.
- **version** (*typing.Union[str, NoneType]*) – Timestamp of collection creation in RFC3339. If this is not provided, the latest version is returned.

Given a collection UUID, return the associated collection object.

classmethod get_file()
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

- **token** (*typing.Union[str, NoneType]*) – Token to manage retries. End users constructing queries should not set this parameter.
- **replica** (*<type 'str'>*) – Replica to fetch from.
- **uuid** (*<type 'str'>*) – A RFC4122-compliant ID for the file.
- **version** (*typing.Union[str, NoneType]*) – Timestamp of file creation in RFC3339. If this is not provided, the latest version is returned.

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()
Retrieve an event subscription given a UUID.

Parameters

- **replica** (*<type 'str'>*) – Replica to fetch from.
- **uuid** (*<type 'str'>*) – A RFC4122-compliant ID for the subscription.

Given a subscription UUID, return the associated subscription.

classmethod get_subscriptions()
Retrieve a user's event subscriptions.

Parameters **replica** (*<type 'str'>*) – Replica to fetch from.

Return a list of associated subscriptions.

classmethod head_file()
Retrieve a file's metadata given an UUID and optionally a version.

Parameters

- **replica** (*<type 'str'>*) – Replica to fetch from.
- **uuid** (*<type 'str'>*) – A RFC4122-compliant ID for the file.
- **version** (*typing.Union[str, NoneType]*) – Timestamp of file creation in RFC3339. 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.

static load_swagger_json(*swagger_json*, *ptr_str=u'$ref'*)
Load the Swagger JSON and resolve {"$ref": "#/..."} internal JSON Pointer references.

login(*access_token=u''*)
Configure and save {prog} authentication credentials.

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

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

classmethod patch_collection()
Update a collection.

Parameters

- **replica** (*<type 'str'>*) – Replica to update the collection on. Updates are propagated to other replicas.
- **uuid** (*<type 'str'>*) – A RFC4122-compliant ID of the collection to update.
- **version** (*<type 'str'>*) – Timestamp of the collection to update in RFC3339 format (required).
- **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.
- **description** (*typing.Union[str, NoneType]*) – New description for the collection.

- **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).
- **details** (*typing.Union[typing.Mapping, NoneType]*) – New details for the 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.

classmethod post_bundles_checkout()

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

Parameters

- **version** (*typing.Union[str, NoneType]*) – Timestamp of file creation in RFC3339. If this is not provided, the latest version is returned.
- **replica** (*<type 'str'>*) – Replica to fetch from.
- **uuid** (*<type 'str'>*) – A RFC4122-compliant ID for the bundle.
- **destination** (*typing.Union[str, NoneType]*) – User-owned destination storage bucket.
- **email** (*typing.Union[str, NoneType]*) – An email address to send status updates to.

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()

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

- **per_page** (*typing.Union[str, NoneType]*) – Max number of results to return per page.
- **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.
- **replica** (*<type 'str'>*) – Replica to search.

- **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.
- **es_query** (*typing.Mapping*) – Elasticsearch query

Accepts Elasticsearch JSON query and returns matching bundle identifiers

Pagination

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.

classmethod put_bundle()

Create a bundle

Parameters

- **version** (*typing.Union[str, NoneType]*) – Timestamp of bundle creation in RFC3339.
- **replica** (*<type 'str'>*) – Replica to write to.
- **uuid** (*<type 'str'>*) – A RFC4122-compliant ID for the bundle.
- **files** (*typing.List*) –
- **creator_uid** (*<type 'int'>*) – User ID who is creating this bundle.

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

classmethod put_collection()

Create a collection.

Parameters

- **replica** (*<type 'str'>*) – Replica to write to.
- **uuid** (*<type 'str'>*) – A RFC4122-compliant ID for the collection.
- **version** (*<type 'str'>*) – Timestamp of collection creation in RFC3339 format. If this is not provided, the version is automatically generated.
- **name** (*<type 'str'>*) – A short name identifying the collection.
- **description** (*<type 'str'>*) – A long description of the collection, formatted in Markdown.
- **contents** (*typing.List*) – A list of objects describing links to files, bundles, other collections, and metadata fragments that are part of the collection.
- **details** (*typing.Mapping*) – Supplementary JSON metadata for the 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.

classmethod put_file()

Create a new version of a file

Parameters

- **version** (*typing.Union[str, NoneType]*) – Timestamp of file creation in RFC3339. If this is not provided, the latest version is returned.
- **uuid** (*<type 'str'>*) – A RFC4122-compliant ID for the file.
- **creator_uid** (*<type 'int'>*) – User ID who is creating this file.
- **source_url** (*<type 'str'>*) – Cloud URL for source data.

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/questions/12186993/what-is-the-algorithm-to-compute-the-amazon-s3-etag-for-a-file-larger-tha
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()

Create a event subscription.

Parameters

- **replica** (*<type 'str'>*) – Replica to write to.
- **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)” }

- **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`.
- **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 corresponding notification HTTP request body will be .. code:: –2769baffc4f24cbc83ced26aa0c2f712 Content-Disposition: form-data; name=”foo” bar Content-Disposition: form-data; name=”payload” {“transaction_id”: “301c9079-3b20-4311-a131-bcda9b7f08ba”, “subscription_id”: … –2769baffc4f24cbc83ced26aa0c2f712– 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.
- **es_query** (*typing.Mapping*) – 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.
- **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`.
- **callback_url** (*<type '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 and no more attemtpts will be made. 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.
- **method** (*typing.Union[str, NoneType]*) – The HTTP request method to use when delivering a notification to the subscriber.

Register an HTTP endpoint that is to be notified when a given event occurs.

upload (*src_dir*, *replica*, *staging_bucket*, *timeout_seconds=1200*)

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.

CHAPTER 7

Table of Contents

Python Module Index

h

Index