Data Object Service Documentation

Release 0.3.0

David Steinberg

Feb 06, 2019

Contents

Welcome to the documentation for the Data Object Service Schemas! These schemas present an easy-to-implement interface for publishing and accessing data in heterogeneous storage environments. It also includes a demonstration client and server to make creating your own DOS implementation easy!

CHAPTER 1

Schemas for the Data Object Service (DOS) API

The Global Alliance for Genomics and Health is an international coalition formed to enable the sharing of genomic and clinical data. This collaborative consortium takes place primarily via GitHub and public meetings.

1.1 Cloud Workstream

The Data Working Group concentrates on data representation, storage, and analysis, including working with platform development partners and industry leaders to develop standards that will facilitate interoperability. The Cloud Workstream is an informal, multi-vendor working group focused on standards for exchanging Docker-based tools and CWL/WDL workflows, execution of Docker-based tools and workflows on clouds, and abstract access to cloud object stores.

1.2 What is DOS?

This proposal for a DOS release is based on the schema work of Brian W. and others from OHSU along with work by UCSC. It also is informed by existing object storage systems such as:

- GNOS (as used by PCAWG)
- ICGC Storage (as used to store data on S3, see overture-stack/score)
- Human Cell Atlas Storage (see HumanCellAtlas/data-store)
- NCI GDC Storage
- Keep by Curoverse (see curoverse/arvados)

The goal of DOS is to create a generic API on top of these and other projects, so workflow systems can access data in the same way regardless of project.

1.3 Key features

1.3.1 Data object management

This section of the API focuses on how to read and write data objects to cloud environments and how to join them together as data bundles. Data bundles are simply a flat collection of one or more files. This section of the API enables:

- create/update/delete a file
- create/update/delete a data bundle
- register UUIDs with these entities (an optionally track versions of each)
- generate signed URLs and/or cloud specific object storage paths and temporary credentials

1.3.2 Data object queries

A key feature of this API beyond creating/modifying/deletion files is the ability to find data objects across cloud environments and implementations of DOS. This section of the API allows users to query by data bundle or file UUIDs which returns information about where these data objects are available. This response will typically be used to find the same file or data bundle located across multiple cloud environments.

1.4 Implementations

There are currently a few experimental implementations that use some version of these schemas.

- DOS Connect observes cloud and local storage systems and broadcasts their changes to a service that presents DOS endpoints.
- DOS Downloader is a mechanism for downloading Data Objects from DOS URLs.
- dos-gdc-lambda presents data from the GDC public REST API using the Data Object Service.
- dos-signpost-lambda presents data from a signpost instance using the Data Object Service.

1.5 More information

- Global Alliance for Genomics and Health
- GA4GH Cloud Workstream

CHAPTER 2

Quickstart

2.1 Installing

Installing is quick and easy. First, it's always good practice to work in a virtualenv:

```
$ virtualenv venv
$ source venv/bin/activate
```

Then, install from PyPI:

```
$ pip install ga4gh-dos-schemas
```

Or, to install from source:

```
$ git clone https://github.com/ga4gh/data-object-service-schemas.git
$ cd data-object-service-schemas
$ python setup.py install
```

2.2 Running the client and server

There's a handy command line hook for the server:

```
$ ga4gh_dos_server
```

and for the client:

```
$ ga4gh_dos_demo
```

(The client doesn't do anything yet but will soon.)

2.3 Further reading

- gdc_notebook.ipynb outlines examples of how to access data with this tool.
- demo.py demonstrates basic CRUD functionality implemented by this package.

CHAPTER 3

Data Object Service Demonstration Server

DOS Demonstration Server

Running this server will start an ephemeral Data Object Service (its registry contents won't be saved after exiting). It uses the connexion module to translate the OpenAPI schema into named controller functions.

These functions are described in *ga4gh.dos.controllers* and are meant to provide a simple implementation of DOS.

Data Object Service Controller Functions

These controller functions for the demo server implement an opinionated version of DOS by providing uuid's to newly create objects, and using timestamp versions.

Initializes an in-memory dictionary for storing Data Objects.

ga4gh.dos.controllers.**CreateDataBundle**(***kwargs*)
Create a Data Bundle, issuing a new identifier if one is not provided.

Parameters **kwargs** –

Returns

ga4gh.dos.controllers.**CreateDataObject**(***kwargs*)
Creates a new Data Object by issuing an identifier if it is not provided.

Parameters **kwargs** –

Returns

ga4gh.dos.controllers.**DeleteDataBundle**(***kwargs*)
Deletes a Data Bundle by ID.

Parameters **kwargs** –

Returns

ga4gh.dos.controllers.**DeleteDataObject**(***kwargs*)
Delete a Data Object by data_object_id.

Parameters **kwargs** –

Returns

ga4gh.dos.controllers.**GetDataBundle**(***kwargs*)
Get a Data Bundle by identifier.

Parameters **kwargs** –

Returns

ga4gh.dos.controllers.**GetDataBundleVersions**(***kwargs*)
Get all versions of a Data Bundle.

Parameters **kwargs** –

Returns

ga4gh.dos.controllers.**GetDataObject**(***kwargs*)
Get a Data Object by data_object_id. :param kwargs: :return:

ga4gh.dos.controllers.**GetDataObjectVersions**(***kwargs*)
Returns all versions of a Data Object. :param kwargs: :return:

ga4gh.dos.controllers.**ListDataBundles**(***kwargs*)
Takes a ListDataBundles request and returns the bundles that match that request. Possible kwargs: alias, url, checksum, checksum_type, page_size, page_token

Parameters **kwargs** – ListDataBundles request.

Returns

ga4gh.dos.controllers.**ListDataObjects**(***kwargs*)
Returns a list of Data Objects matching a ListDataObjectsRequest.

Parameters **kwargs** – alias, url, checksum, checksum_type, page_size, page_token

Returns

ga4gh.dos.controllers.**UpdateDataBundle**(***kwargs*)
Updates a Data Bundle to include new metadata by upserting the new bundle.

Parameters **kwargs** –

Returns

ga4gh.dos.controllers.**UpdateDataObject**(***kwargs*)
Update a Data Object by creating a new version.

Parameters **kwargs** –

Returns

ga4gh.dos.controllers.**add_created_timestamps**(*doc*)
Adds created and updated timestamps to the document. :param doc: A document to be timestamped :return doc: The timestamped document

ga4gh.dos.controllers.**add_updated_timestamps**(*doc*)
Adds created and updated timestamps to the document.

ga4gh.dos.controllers.**create**(*body*, *key*)
Creates a new document at the given key by adding necessary metadata and storing in the in-memory store. :param body: :param key: :return:

ga4gh.dos.controllers.**filter_data_bundles**(*predicate*)
Filters data bundles according to a function that acts on each item returning either True or False per item. :param predicate: A function used to test items :return: List of Data Bundles

ga4gh.dos.controllers.**filter_data_objects**(*predicate*)
Filters data objects according to a function that acts on each item returning either True or False per item.

ga4gh.dos.controllers.**get_most_recent**(*key*)
Gets the most recent Data Object for a key. :param key: :return:

ga4gh.dos.controllers.**get_most_recent_bundle**(*key*)
Returns the most recent bundle for the given key.

Parameters **key** –

Returns

ga4gh.dos.controllers.**now**()
Returns the current time in string format. :return: Current ISO time.

CHAPTER 4

DOS Python HTTP Client

This module exposes a single class *ga4gh.dos.client.Client*, which exposes the HTTP methods of the Data Object Service as named Python functions.

This makes it easy to access resources that are described following these schemas, and uses bravado to dynamically generate the client functions following the OpenAPI schema.

It currently assumes that the service also hosts the swagger.json, in a style similar to the demonstration server, *ga4gh.dos.server*.

class ga4gh.dos.client.**Client**(*url*, *config={'validate_requests': True, 'validate_responses': True}*, *http_client=None*, *request_headers=None*)

This class is the instantiated to create a new connection to a DOS. It connects to the service to download the swagger.json and returns a client in the DataObjectService namespace.

```
from ga4gh.dos.client import Client
client = Client("http://localhost:8000/ga4gh/dos/v1")

models = client.models
c = client.client

# Will return a Data Object by identifier
c.GetDataObject(data_object_id="abc").result()

# To access models in the Data Object Service namespace:
ListDataObjectRequest = models.get_model('ListDataObjectsRequest')

# And then instantiate a request with our own query:
my_request = ListDataObjectsRequest(alias="doi:10.0.1.1/1234")

# Finally, send the request to the service and evaluate the response.
c.ListDataObjects(body=my_request).result()
```

The class accepts a configuration dictionary that maps directly to the bravado configuration.

For more information on configuring the client, see bravado documentation.

classmethod config(*url*, *http_client=None*, *request_headers=None*)
Accepts an optionally configured requests client with authentication details set.

Parameters

- **url** – The URL of the service to connect to
- **http_client** – The http_client to use, defaults to `RequestsClient()`
- **request_headers** – The headers to set on each request.

Returns

ga4gh.dos.client.**validate_int64**(*test*)
Accepts an int64 and checks for numerality. Throws a Swagger Validation exception when failing the test.

Parameters **test** –

Returns

Raises **SwaggerValidationError** –

CHAPTER 5

Contributor's Guide

5.1 Installing

To install for development, install from source (and be sure to install the development requirements as well):

```
$ git clone https://github.com/ga4gh/data-object-service-schemas.git
$ cd data-object-service-schemas
$ python setup.py develop
$ pip install -r requirements.txt
```

5.2 Documentation

We use Sphinx for our documentation. You can generate an HTML build like so:

```
$ cd docs/
$ make html
```

You'll find the built documentation in `docs/build/`.

5.3 Tests

To run tests:

```
$ nosetests python/
```

The Travis test suite also tests for PEP8 compliance (checking for all errors except line length):

```
$ flake8 --select=E121,E123,E126,E226,E24,E704,W503,W504 --ignore=E501 python/
```

5.4 Schema architecture

The canonical, authoritative schema is located at `openapi/data_object_service.swagger.yaml`. All schema changes must be made to the Swagger schema, and all other specifications (e.g. SmartAPI, OpenAPI 3) are derived from it.

5.4.1 Building documents

The schemas are editable as OpenAPI 2 YAML files. To generate OpenAPI 3 descriptions install swagger2openapi and run the following:

```
$ swagger2openapi -y openapi/data_object_service.swagger.yaml > openapi/data_object_
↪service.openapi.yaml
```

5.5 Code contributions

We welcome code contributions! Feel free to fork the repository and submit a pull request. Please refer to this contribution guide for guidance as to how you should submit changes.

CHAPTER 6

Indices and tables

- genindex
- modindex
- search

Python Module Index

g

Index