DataGator Specification: RESTful API v2*

author	LIU Yu <liuyu@opencps.net>
revision	draft-06
license	CC BY-NC-ND 4.0

1. Introduction

The RESTful API of DataGator is a JSON-based programming interface for accessing and manipulating DataGator's computing infrastructure. This document specifies web service endpoints and protocols for invoking the RESTful API of DataGator. Targeted readers of this document are developers experienced in web programming, esp., consuming web services through HTTP messages as specified in RFC 7231.

1.1. Requirements

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in RFC 2119.

2. Overview

API version

This document describes the v2 version of DataGator's RESTful API. All API calls are over HTTPS. Service endpoints are absolute or relative URI templates as defined in RFC 6570. Unless otherwise specified, a relative service endpoint resolves to,

```
https://api.datagator.org/v2{endpoint}
```

JSON schema

Data sent and received through API calls are HTTP messages with JSON objects as payload, which all conform to the draft-4 JSON schema available at,

```
https://api.datagator.org/v2/schema#
```

For example, sending a **GET** request to the *root endpoint* (/) will receive an HTTP response with a *Message* object as its payload, e.g.,

```
GET /v2/ HTTP/1.1
Host: api.datagator.org
```

```
HTTP/1.1 200 OK
Content-Type: application/json
Date: Wed, 03 Jun 2015 14:13:58 GMT
X-DataGator-Entity: Status
X-RateLimit-Limit: 200
X-RateLimit-Remaining: 188
X-RateLimit-Reset: 1433342782

{
    "kind": "datagator#Status",
    "code": 200,
    "version": "v2",
    "service": "datagator.wsgi.api"
}
```

HTTP authentication

Most service endpoints optionally perform client authentication and respond with personalized information matching each client's privileges. Some other resources are dedicated to authenticated clients with matching permissions[1]. Clients are RECOMMENDED to preemptively send the `Authorization` HTTP header when making API calls. The `v2` API supports the following two HTTP authentication schemes.

Basic authentication: Per RFC 7617, the client `credentials` sent with the HTTP request is the concatenation of the `username`, a single colon (`":"`) character, and the `password`, encoded into a string of ASCII characters using Base64.

```
GET /v2/ HTTP/1.1
Host: api.datagator.org
Authorization: Basic {credentials}
```

Token authentication: The `access_token` sent with the HTTP request is an opaque string of ASCII characters issued to the client.

```
GET /v2/ HTTP/1.1
Host: api.datagator.org
Authorization: Token {access_token}
```

HTTP redirection

API uses HTTP redirections where appropriate. Receiving an HTTP redirection is *not* an error, and clients SHOULD follow the redirection by default. Redirect responses will have a `Location` header containing the URI of the targeted resource.

pagination

Some *listing* service endpoints return a *paginated list* of entities encapsulated in a *Page* object. HTTP **`GET`** requests to these services can take an optional `?page` parameter in the query string to specify the *zero*-based page number of interest. A *Page* object contains 10 to 20 items by default. For some resources, the size of a *Page* object can be customized to contain up to 100 items with a `?page_size` parameter.

```
GET /v2/repo/Pardee/IGOs/data/?page=2&page_size=30 HTTP/1.1
Host: api.datagator.org
Accept: */*
```

```
HTTP/1.1 200 OK
Content-Type: application/json
Date: Fri, 04 Sep 2015 06:16:41 GMT
Link: </v2/repo/Pardee/IGOs.1/data?page=0&page_size=30>; rel="first",
    </v2/repo/Pardee/IGOs.1/data?page=1&page_size=30>; rel="prev",
    </v2/repo/Pardee/IGOs.1/data?page=3&page_size=30>; rel="next",
    </v2/repo/Pardee/IGOs.1/data?page=13&page_size=30>; rel="last"
```

[1] To prevent accidental leakage of private information, some service endpoints will return `404 Not Found`, instead of `403 Forbidden`, to unauthorized clients.

```
X-RateLimit-Limit: 200
X-RateLimit-Remaining: 198
X-RateLimit-Reset: 1441350986

{
    "kind": "datagator#Page",
    "items": [
        {"kind": "datagator#Matrix", "name": "ASEF"},
        {"kind": "datagator#Matrix", "name": "ASPAC"},
        ...
        {"kind": "datagator#Matrix", "name": "CBI"}
    ],
    "startIndex": 60,
    "itemsPerPage": 30,
    "itemsCount": 20
}
```

Service endpoints that return *Page* objects MAY also provide RFC 5988 `Link` headers containing one or more of the following link relations.

Relation	Description
`first`	link to the initial *Page*
`prev`	link to the immediate previous *Page*
`next`	link to the immediate next *Page*
`last`	link to the last non-empty *Page*

When enumerating a *paginated* resource, clients are recommended to follow the `Link` relations instead of constructing URIs by themselves. Note that, the *pagination* of resource is open-ended. Querying a `?page` number beyond the `last` page is *not* an error, and will receive an empty *Page* object, instead of `404 Not Found`.

rate limiting

Authenticated clients can make up to 2,000 API calls per hour. For unauthorized clients, the limit is 200 calls per hour and is associated with the client's' IP address. The rate limit status is included in the `X-RateLimit-*` headers of HTTP responses.

HTTP Header	Description
`X-RateLimit-Limit`	The hourly limit of API calls allowed for the current client.
`X-RateLimit-Remaining`	The number of API calls remaining in current rate limit window.
`X-RateLimit-Reset`	The UNIX time at which the current rate limit window resets.

Exceeding the rate limit will receive a `429 Too Many Requests` response.

```
HTTP/1.1 429 TOO MANY REQUESTS
Content-Type: application/json
Date: Fri, 05 Sep 2015 03:10:56 GMT
X-DataGator-Entity: Error
X-RateLimit-Limit: 200
X-RateLimit-Remaining: 0
X-RateLimit-Reset: 1441426202
Content-Length: 110
Retry-After: 3546

{
    "kind": "datagator#Error",
    "code": 429,
```

```
    "service": "datagator.rest.api",
    "message": "API request over-rate."
}
```

Note that, invoking some *expensive* services, such as *full-text search* and *dataset revision*, may be counted as multiple API calls.

CORS

The `v2` API accepts client-side requests from any origin.

```
GET /v2/ HTTP/1.1
Host: api.datagator.org
Origin: http://example.com
```

```
HTTP/1.1 200 OK
Content-Type: application/json
Date: Fri, 16 Oct 2015 13:25:57 GMT
Access-Control-Allow-Credentials: true
Access-Control-Allow-Methods: GET, HEAD, POST, PUT, PATCH, DELETE
Access-Control-Allow-Origin: http://example.com
Access-Control-Expose-Headers: ETag, Last-Modified, Link, Location,
    X-RateLimit-Limit, X-RateLimit-Remaining, X-RateLimit-Reset,
    X-DataGator-Entity
X-DataGator-Entity: Status
Vary: Origin
```

Know Issues: Due to a known issue of the web server being used by the backend system, `304 Not Modified` responses do *not* currently contain CORS headers.

3. Repository Service

3.1. Repo Base Endpoint

`/repo/{repo}{?filter}`

`GET`: get the *Repo* object identified by `repo`.

Variable	Flag (default)	Description
filter	active (+)	include active *DataSets*
	hidden (-)	include in-active *DataSets*
	public (+)	include public *DataSets*
	protected (+)	include non-public *DataSets*

On success, the response is a *Repo* object, e.g.,

```
GET /v2/repo/Pardee?filter=-hidden HTTP/1.1
Host: api.datagator.org
```

```
HTTP/1.1 200 OK
Content-Type: application/json
Cache-Control: private
Date: Wed, 22 Jul 2015 00:09:31 GMT
X-DataGator-Entity: Repo
Link: </v2/repo/Pardee/>; rel="contents"

{
```

```
    "kind": "datagator#Repo",
    "name": "Pardee",
    "itemsCount": 2,
    "size": 51127
}
```

access control

The aggregated statistics of the *Repo* object represent *DataSets* that (i) satisfy the `?filter` parameter, and (ii) match the permissions available to the client. Namely, there could be other *non-public DataSets* in the same *Repo* that are *not* reflected in the statistics, because the client does *not* have matching permissions.

errors

On failure, the response is a *Message* object with error code and description, e.g.,

```
GET /v2/repo/NonExistence HTTP/1.1
Host: api.datagator.org
```

```
HTTP/1.1 404 NOT FOUND
Content-Type: application/json

{
    "kind": "datagator#Error",
    "code": 404,
    "message": "Invalid repository 'NonExistence'",
    "service": "datagator.rest.api"
}
```

3.2. Repo Content Endpoint

`/repo/{repo}/{?filter,order}`

GET: get a paginated list of *DataSets* from the *Repo* object identified by `repo`.

Variable	Flag (default)	Description
filter	active (+)	include active *DataSets*
	hidden (-)	include in-active *DataSets*
	public (+)	include public *DataSets*
	protected (+)	include non-public *DataSets*
order	name	default: `order=-updated`
	size	
	updated	

3.3. DataSet Base Endpoint

`/repo/{repo}/{dataset}{?filter}`

`/repo/{repo}/{dataset}{.rev}{?filter}`

GET: get the *DataSet* object identified by `repo/dataset`. If the URI specifies the `.rev` suffix, returns the *historical* revision identified by `rev`; otherwise, returns the *latest* revision (aka. `HEAD`) of the *DataSet*.

Variable	Flag (default)	Description
filter	Matrix (+)	include *Matrix* items
	Recipe (+)	include *Recipe* items
	Opaque (-)	include *Opaque* items

On success, the response is the targeted *DataSet* object, e.g.,

```
GET /v2/repo/Pardee/Nodes?filter=-Matrix HTTP/1.1
Host: api.datagator.org
```

```
HTTP/1.1 200 OK
Content-Type: application/json
Date: Fri, 20 Nov 2015 10:30:38 GMT
Last-Modified: Sat, 14 Nov 2015 01:52:08 GMT
ETag: "d31cf201a01a54efbecc9482dd2e1616"
Link: </v2/repo/Pardee/Nodes.1/data>; rel="contents"
X-DataGator-Entity: DataSet

{
    "kind": "datagator#DataSet",
    "name": "Nodes",
    "repo": {
        "kind": "datagator#Repo",
        "name": "Pardee"
    },
    "rev": 1,
    "created": "2015-11-13T19:14:22Z",
    "createdBy": {
        "kind": "datagator#User",
        "name": "Pardee",
        "displayName": null,
        "public": false,
        "joined": "2015-11-13T18:46:35Z"
    },
    "updated": "2015-11-14T01:52:08Z",
    "updatedBy": {
        "kind": "datagator#User",
        "name": "Pardee",
        "displayName": null,
        "public": false,
        "joined": "2015-11-13T18:46:35Z"
    },
    "public": false,
    "active": true,
    "itemsCount": 25,
    "size": 38069
}
```

PUT: create or update the *DataSet* identified by `repo/dataset`. It is RECOMMENDED that the URI does *not* specify the `.rev` suffix; if otherwise specified, `rev` SHOULD be a *valid* revision of the targeted *DataSet*. The request body SHOULD be a valid *DataSet* object satisfying the following expectations.

Property[2]	Expectation
/repo	valid *Repo* object
/repo/name	consistent with `repo` in the URI
/name	consistent with `dataset` in the URI
/public	optional and defaults to `false` when creating a new *DataSet*; required otherwise

access control

PUT is a *committal* operation requiring authentication.

On success, the response is a *Message* object with status code `201 Created`, or `200 Ok`, depending whether the targeted *DataSet* was *created* or *updated*.

```
PUT /v2/repo/Pardee/SampleData HTTP/1.1
Host: api.datagator.org
Authorization: Basic ************
Content-Type: application/json

{
    "kind": "datagator#DataSet",
    "repo": {
        "kind": "datagator#Repo",
        "name": "Pardee"
    },
    "name": "SampleData"
}
```

```
HTTP/1.1 201 CREATED
Date: Fri, 20 Nov 2015 15:14:00 GMT
X-DataGator-Entity: Status
Content-Type: application/json

{
    "kind": "datagator#Status",
    "code": 201,
    "service": "datagator.rest.api",
    "message": "Created dataset."
}
```

DELETE: inactivate the *DataSet* identified by `repo/dataset`. The request body SHOULD be empty.

3.4. DataSet Content Endpoint

`/repo/{repo}/{dataset}/data`

`/repo/{repo}/{dataset}{.rev}/data/`

GET: get a paginated list of *DataItems* from the *DataSet* identified by `repo/dataset`. If the URI specifies the `.rev` suffix, returns the content of the *historical* revision identified by `rev`; otherwise, returns that of the *latest* revision (aka. `HEAD`) of the *DataSet*.

Variable	Flag (default)	Description
filter	Matrix (+)	include *Matrix* items
	Recipe (+)	include *Recipe* items
	Opaque (-)	include *Opaque* items
order	name	default: `order=name`
	kind	
	mediaType	
	size	
	flag	

[2] Properties are RFC 6901 JSON pointers w.r.t. the request body.

`PATCH:` commit a new *revision* to the *DataSet*. The request body SHOULD be a *DataSet* object satisfying the following expectations.

Property	Expectation
/repo	valid *Repo* object
/repo/name	consistent with `repo` in the URI
/name	consistent with `dataset` in the URI
/items/0 ⋮ /items/(n-1)	`/items/i` ($i = 0,\ldots,n-1$) SHOULD be valid *DataItem* objects with `kind`, `name`, and `data` properties. Each element of `/items` specifies one of the three *operations* as follows, **create**: if (i) `data` is *not* `null`, and (ii) the `HEAD` revision of the *DataSet* does *not* contain the *DataItem* identified by `kind` and `name`, then, the pending revision will incorporate a new *DataItem* with `data` as its content. **update**: if (i) `data` is *not* `null`, and (ii) the `HEAD` revision of the *DataSet* already contains the *DataItem* identified by `kind` and `name`, then, in the pending revision the content of the *DataItem* will be replaced with `data`. **delete**: if (i) `data` is `null`, and (ii) the `HEAD` revision of the *DataSet* contains the *DataItem* identified by `kind` and `name`, then the *DataItem* will be eliminated in the pending revision[3].
/itemsCount	consistent with the # of elements in `/items`

bulk operation

The request body MAY submit one or more of the above-mentioned *operations*. For instance, the following request will (i) **create** or **update** the *Matrix* named `UN`, and (ii) **delete** the *Recipe* named `Sum.recipe`. All other *DataItems* that exist in the `HEAD` revision of the targeted *DataSet* but are missing from the request body will be preserved *as-is* in the pending revision.

```
PATCH /v2/repo/Pardee/IGO_Members/data HTTP/1.1
Host: api.datagator.org
Authorization: Basic ************
Content-Type: application/json

{
    "kind": "datagator#DataSet",
    "repo": {
        "kind": "datagator#Repo",
        "name": "Pardee"
    },
    "name": "IGO_Members",
    "items": [
        {
            "kind": "datagator#Matrix",
            "name": "UN",
            "data": {
                "kind": "datagator#Matrix",
                "columnHeaders": 1,
```

[3] If, otherwise, the `HEAD` revision does *not* contain the *DataItem*, then the **delete** operation itself will be ignored, thus not affecting the pending revision.

```
                "rowHeaders": 1,
                "rows": [
                    ["Country", 2010, 2011, 2012, 2013, 2014],
                    ["China", None, 1, None, 1, 1],
                    ["United States", None, 1, 1, None, 1]
                ],
                "rowsCount": 3,
                "columnsCount": 6
            }
        },
        {
            "kind": "datagator#Recipe",
            "name": "Sum",
            "data": null
        }
    ],
    "itemsCount": 2
}
```


asynchronous revision

PATCH is an *asynchronous* operation. On success, the response is a *Message* object with status code `202 Accepted`. Such a response indicates that the submitted *revision* has passed preliminary check and is scheduled for execution in an asynchronous *Task*. By the time of response, the *Task* is *not* guaranteed to complete, or succeed at all. Clients are RECOMMENDED to poll the status of the *Task* via the URI available in the `Location` header of the response.


```
HTTP/1.1 202 ACCEPTED
Location: https://api.datagator.org/v2
    /task/c6266af4-d4fa-4764-8481-b189c1dfe999
Content-Type: application/json

{
    "kind": "datagator#Status",
    "code": 202,
    "service": "datagator.wsgi.api",
    "message": "Scheduled dataset revision."
}
```


atomicity

Remarks: *DataSet revision* is *atomic*. All *operations* submitted in the same **PATCH** request will be committed in a single transaction by the *asynchronous Task*. If any of the *operations* fails, then the *revision* will be revoked entirely, and the `HEAD` revision of the targeted *DataSet* will remain intact. In case a **PATCH** request contains conflicting *operations* on the same *DataItem* -- e.g., both **update** and **delete**, or multiple **updates** with distinct `data` -- the *Task* MAY still succeed, but the outcome is *undefined*. In addition, if the **PATCH** request is *trivial* -- i.e., the enclosed *operations* not yielding effective changes to the `HEAD` revision of the targeted *DataSet*, such as (i) **update** operations with `data` identical to those found in the `HEAD` revision, (ii) **delete** operations targeting non-existent *DataItems* -- then the pending revision will *not* be committed.

3.5. DataItem Content Endpoint

`/repo/{repo}/{dataset}/data/{key}{?format}`

`/repo/{repo}/{dataset}{.rev}/data/{key}{?format}`

GET: get the content of the *DataItem* identified by `key` from the container *DataSet* identified by `repo/dataset`. If the URI specifies the `.rev` suffix, returns the *DataItem* content as of the *historical* revision identified by `rev`; otherwise, returns the *DataItem* content as of the *latest* revision (aka. `HEAD`) of the *DataSet*.

content negotiation

GET supports *content negotiation* for *Matrix* and *Recipe* items, either through (i) the `Accept` header of the request, or (ii) the `?format` parameter. The mappings between *media types* and `?format` values are summarized as follows.

Accept Header (*Matrix*)	**Format**
`application/vnd.datagator.matrix+json`	`json`
`application/json`	
`application/vnd.openxmlformats-officedocument.spreadsheetml.sheet`	`xlsx`

Accept Header (*Recipe*)	**Format**
`application/vnd.datagator.recipe+json`	`json`
`application/json`	
`application/vnd.datagator.recipe+dgml`	`dgml`

Note that *Opaque* items do *not* support *content negotiation*. As the name suggests, the content of an *Opaque* item is archived *as-is* by the backend system, and is thus only available in the original format specified at creation.

Accept Header (*Opaque*)	**Format**
`*/*`	N/A

access control

The content of a *DataItem* is restricted to *authenticated* clients with `read` permissions on the container *DataSet*.

```
GET /v2/repo/Pardee/Nodes/data/Append.recipe HTTP/1.1
Host: api.datagator.org
Authorization: Basic ************
Accept: application/vnd.datagator.recipe+json
```

```
HTTP/1.1 200 OK
Date: Wed, 02 Dec 2015 14:24:42 GMT
X-DataGator-Entity: Recipe
Last-Modified: Sat, 14 Nov 2015 01:52:08 GMT
ETag: "5ac9509ad4b8573a92ae8bcbbebb6220"
Link: </v2/repo/Pardee/Nodes2/data/Append>; rel="related"
Content-Type: application/json

{
    "kind": "datagator#Recipe",
    ...
}
```

conditional request

To facilitate client-side cache control, the request MAY specify RFC 7232 *conditional request* headers `If-None-Match` or `If-Modified-Since`, e.g.,

```
GET /v2/repo/Pardee/Nodes2/data/Append.recipe HTTP/1.1
Host: api.datagator.org
Authorization: Basic ************
Accept: application/vnd.datagator.recipe+json
If-None-Match: "5ac9509ad4b8573a92ae8bcbbebb6220"
```

```
HTTP/1.1 304 NOT MODIFIED
Date: Wed, 02 Dec 2015 14:40:46 GMT
ETag: "5ac9509ad4b8573a92ae8bcbbebb6220"
```

Remarks: The content of a *DataItem* can be considerably large in size. Clients are RECOMMENDED to cache the content and use conditional requests whenever possible to avoid repetitive transmission.

PUT: create / update the *DataItem* object identified by `key` from the container *DataSet* identified by `repo/dataset`. The request body SHOULD be a valid content object for the *DataItem*. **PUT** is a *committal* operation requiring authentication.

acess control

On success, the response is the *DataItem* object with status code `201 Created`, or `200 Ok`, depending on whether its content was *created* or *updated*.

```
PUT /v2/repo/Pardee/Nodes2/data/Append HTTP/1.1
Host: api.datagator.org
Authorization: Basic ************
Content-Type: application/json

{
    "kind": ""datagator#Matrix",
    "columnHeaders": 1,
    "rowHeaders": 1,
    "rows": [
        ["Country", 2010, 2011, 2012, 2013, 2014],
        ["China", None, 1, None, 1, 1],
        ["United States", None, 1, 1, None, 1]
    ],
    "rowsCount": 3,
    "columnsCount": 6
}
```

```
HTTP/1.1 201 CREATED
X-DataGator-Entity: Status
ETag: "a0211b7c07cb5f058caca9a62d853181"
Content-Type: application/json
Location: http://api.datagator.org/v2
    /repo/Pardee/Nodes2/data/Append

{
    "kind": "datagator#Matrix",
    "name": "Append",
    "mediaType": null,
    "digest": "84a41b117476cb8f59a85175c2858c34
        a4ff9b64d60f73e5b1092999ab359f20",
    "flag": "C",
    "created": "2015-12-02T15:19:43Z",
    "createdBy": {
        "kind": "datagator#User",
        "name": "liuyu",
        "displayName": null,
        "public": false,
        "joined": "2015-11-16T14:19:04Z"
    },
    "updated": "2015-12-02T15:19:43Z",
    "updatedBy": {
        "kind": "datagator#User",
        "name": "liuyu",
        "displayName": null,
        "public": false,
        "joined": "2015-11-16T14:19:04Z"
    },
    "size": 213
}
```