WaterButler

Release 23.0.0

Aug 18, 2023

Contents

1 Quick links **3**

2 Documentation **5**
- 2.1 Getting Started . . . 5
- 2.2 Overview . . . 6
- 2.3 API . . . 7
- 2.4 Providers . . . 15
- 2.5 Adding A New Provider . . . 16
- 2.6 Rate-limiting . . . 16
- 2.7 Code . . . 17
- 2.8 Releases . . . 32
- 2.9 Testing . . . 32

3 Project info **35**
- 3.1 Contributing . . . 35
- 3.2 ChangeLog . . . 36
- 3.3 License . . . 58

Python Module Index **63**

Index **65**

WaterButler is a Python web application for interacting with various file storage services via a single RESTful API, developed at The Center for Open Science.

CHAPTER 1

Quick links

- Source (github)

CHAPTER 2

Documentation

This documentation is also available in PDF and Epub formats.

2.1 Getting Started

2.1.1 Setting Up

Make sure that you are using >= python3.5 and install invoke for your current python3 version.

```
pip install setuptools==37.0.0
pip install invoke==0.13.0
```

Install requirements

```
invoke install
```

Or for some nicities (like tests)

```
invoke install --develop
```

Start the server

```
invoke server
```

Start the celery worker

```
invoke celery
```

2.1.2 Contributing

See CONTRIBUTING.md.

2.1.3 Known Issues

Updated, 2018-01-02: *WB has been updated to work with setuptools==37.0.0, as of WB release v0.37. The following issue should not happen for new installs, but may occur if you downgrade to an older version.* Running `invoke install -d` with setuptools v31 or greater can break WaterButler. The symptom error message is: `"AttributeError: module 'waterbutler' has no attribute '__version__'"`. If you encounter this, you will need to remove the file `waterbutler-nspkg.pth` from your virtualenv directory, run `pip install setuptools==30.4.0`, then re-run `invoke install -d`.

`invoke $command` results in `'$command' did not receive all required positional arguments!`: this error message occurs when trying to run WaterButler v0.30.0+ with `invoke<0.13.0`. Run `pip install invoke==0.13.0`, then retry your command.

2.1.4 Running Tests

Make sure that you already have dev-requirements

```
invoke test
```

2.2 Overview

WaterButler is a Python web application for interacting with various file storage services via a single RESTful API.

Authentication to WB can be done with HTTP Basic Authorization or via cookies.

Request lifecycle: User makes request, credentials are requested from auth provider, request is made to storage provider, response is returned to user:

```
User                   WaterButler                                    OSF    Provider
 |                          |                                          |        |
 | -- request resource ---> |                                          |        |
 |                          |                                          |        |
 |                          | --- request credentials for provider --> |        |
 |                          |                                          |        |
 |                          | <-- return credentials, if valid user -- |        |
 |                          |                                                   |
 |                          | ----- relay user request to provider w/ creds --> |
 |                          |                                                   |
 |                          | <---- return request response to WB ------------- |
 |                          |
 | <-- provider response -- |
```

If the user is interacting with WaterButler via the OSF, the diagram looks like this:

```
User                       OSF                                WaterButler                ␣
↪       Provider
 |                          |                                      |                    ␣
↪          |
 | -- request resource ---> |                                      |                    ␣
↪          |
 |                          |                                      |                    ␣
↪          |
 |                          | --- relay user req. --------------> |                    ␣
↪          |
```

(continues on next page)

(continued from previous page)

```
 |                          |                                    |                 ␣
↪          |
 |                          | <-- request creds for provider ---- |                 ␣
↪          |
 |                          |                                    |                 ␣
↪          |
 |                          | --- return creds, if valid user --> |                 ␣
↪          |
 |                          |                                    |                 ␣
↪          |
 |                          |                                    | --- relay req. w/␣
↪creds --> |
 |                          |                                    |                 ␣
↪          |
 |                          |                                    | <-- return resp. -
↪--------- |
 |                          |                                    |
 |                          | <-- relay provider resp. to OSF --- |
 |                          |
 | <-- provider response -- |
```

Only one auth provider so far, the OSF.

Two APIs, v0 and v1. v0 is deprecated.

2.2.1 Terminology

auth provider - The service that provides the authentication needed to communicate with the storage provider. WaterButler currently only supports the Open Science Framework as an auth provider.

storage provider - The external service being connected to. ex. Google Drive, GitHub, Dropbox.

provider - When we refer to a *provider* without specifying which type, we are talking about a *storage provider*.

resource - The parent resource the provider is connected to. This will depend on the auth provider. For the OSF, the resource is the GUID of the project that the provider is connected to. For example, the OSF project for the Reproducibility Project: Psychology is found at https://osf.io/ezcuj/. The *resource* in this case is `ezcuj`. When a request is made to WaterButler for something under the `ezcuj` resource, a query will be sent to the OSF to make sure the authenticated user has permission to access the provider linked to that project.

2.3 API

2.3.1 v0 API

> **Warning:** The v0 WaterButler API is deprecated and should no longer be used. It is only documented to provide a reference for legacy consumers.

TODO: v0 api docs

2.3.2 v1 API

The version 1 WaterButler API tries to conform to RESTful principles. A v1 url takes the form:

```
http://files.osf.io/v1/resources/<node_id>/providers/<provider_id>/<id_or_path>

e.g. http://files.osf.io/v1/resources/jy4bd/providers/osfstorage/523402a0234
```

Here `jy4bd` is the id of an OSF project, `osfstorage` is the provider, and `523402a0234` is the identifier of a particular file.

2.3.3 Conventions

Trailing slashes are significant. When `<id_or_path>` refers to a folder, it must *always* have a trailing slash. If it's a file, it must *never* have a trailing slash. Some providers allow files and folders to have the same name within a directory. The slash indicates user intention. This is true even for providers that use IDs. If the request URL contains the ID `1a23490d777c/` but `1a23490d777c` refers to a file, WB will return a 404 Not Found.

Create returns 201, Update returns 200, Move/Copy returns depends. A successful file or folder creation operations should always return a 201 Created status (or 202 Accepted). A successful update/rename operation should always return a 200 Updated. Move / copy should return 200 if overwriting a file at the destination path, otherwise it should return 200.

2.3.4 Actions

The links property of the response provides endpoints for common file operations. The currently-supported actions are:

Get Info (files, folders)

```
Method:   GET
Params:   ?meta=
Success:  200 OK + file representation
Example:  GET /resources/mst3k/providers/osfstorage/?meta=
```

The contents of a folder or details of a particular file can be retrieved by performing a GET request against the entity's URL with the `meta=` query parameter appended. The response will be a JSON-API formatted response.

Download (files)

```
Method:   GET
Params:   <none>
Success:  200 OK + file body
Example:  GET /resources/mst3k/providers/osfstorage/2348825492342
```

To download a file, issue a GET request against its URL. The response will have the Content-Disposition header set, which will will trigger a download in a browser.

Download Zip Archive (folders)

```
Method:   GET
Params:   <none>
Success:  200 OK + folder body
Example:  GET /resources/mst3k/providers/osfstorage/23488254123123/?zip=
```

To download a zip archive of a folder, issue a GET request against its URL. The response will have the Content-Disposition header set, which will will trigger a download in a browser.

Create Subfolder (folders)

```
Method:       PUT
Query Params: ?kind=folder&name={new_folder_name}
Body:         <empty>
Success:      201 Created + new folder representation
Example:      PUT /resources/mst3k/providers/osfstorage/?kind=folder&name=foo-folder
```

You can create a subfolder of an existing folder by issuing a PUT request against the new_folder link. The `?kind=folder` portion of the query parameter is already included in the new_folder link. The name of the new subfolder should be provided in the name query parameter. The response will contain a WaterButler folder entity. If a folder with that name already exists in the parent directory, the server will return a 409 Conflict error response.

Upload New File (folders)

```
Method:       PUT
Query Params: ?kind=file&name={new_file_name}
Body (Raw):   <file data (not form-encoded)>
Success:      201 Created + new file representation
Example:      PUT /resources/mst3k/providers/osfstorage/?kind=file&name=foo-file
```

To upload a file to a folder, issue a PUT request to the folder's upload link with the raw file data in the request body, and the kind and name query parameters set to 'file' and the desired name of the file. The response will contain a WaterButler file entity that describes the new file. If a file with the same name already exists in the folder, the server will return a 409 Conflict error response. The file may be updated via the url in the create response's `/links/upload` attribute.

Update Existing File (file)

```
Method:       PUT
Query Params: ?kind=file
Body (Raw):   <file data (not form-encoded)>
Success:      200 OK + updated file representation
Example:      PUT /resources/mst3k/providers/osfstorage/2348825492342?kind=file
```

To update an existing file, issue a PUT request to the file's upload link with the raw file data in the request body and the kind query parameter set to "file". The update action will create a new version of the file. The response will contain a WaterButler file entity that describes the updated file.

Rename (files, folders)

```
Method:        POST
Query Params:  <none>
Body (JSON):   {
                 "action": "rename",
                 "rename": {new_file_name}
               }
Success:       200 OK + new entity representation
```

To rename a file or folder, issue a POST request to the move link with the action body parameter set to "rename" and the rename body parameter set to the desired name. The response will contain either a folder entity or file entity with the new name.

Move & Copy (files, folders)

```
Method:        POST
Query Params:  <none>
Body (JSON):   {
                 // mandatory
                 "action":   "move"|"copy",
                 "path":     {path_attribute_of_target_folder},
                 // optional
                 "rename":   {new_name},
                 "conflict": "replace"|"keep"|"warn", // defaults to 'warn'
                 "resource": {node_id},               // defaults to current {node_id}
                 "provider": {provider}               // defaults to current {provider}
               }
Success:       200 OK or 201 Created + new entity representation
```

Move and copy actions both use the same request structure, a POST to the move url, but with different values for the action body parameters. The path parameter is also required and should be the OSF path attribute of the folder being written to. The rename and conflict parameters are optional. If you wish to change the name of the file or folder at its destination, set the rename parameter to the new name. The conflict param governs how name clashes are resolved. Possible values are `replace`, `keep`, and `warn`. `warn` is the default and will cause WaterButler to throw a 409 Conflict error if the file that already exists in the target folder. `replace` will tell WaterButler to overwrite the existing file, if present. `keep` will attempt to keep both by adding a suffix to the new file's name until it no longer conflicts. The suffix will be ' (x)' where x is a increasing integer starting from 1. This behavior is intended to mimic that of the OS X Finder. The response will contain either a folder entity or file entity with the new name.

Files and folders can also be moved between nodes and providers. The resource parameter is the id of the node under which the file/folder should be moved. It must agree with the path parameter, that is the path must identify a valid folder under the node identified by resource. Likewise, the provider parameter may be used to move the file/folder to another storage provider, but both the resource and path parameters must belong to a node and folder already extant on that provider. Both resource and provider default to the current node and providers.

The return value for a successful move or copy will be the metadata associated with the file or in the case of foldersm the metadata associated with that folder and its immediate children.

If a moved/copied file is overwriting an existing file, a 200 OK response will be returned. Otherwise, a 201 Created will be returned.

Delete (file, folders)

```
Method:        DELETE
Query Params:  ?confirm_delete=1 // required for root folder delete only
Success:       204 No Content
```

To delete a file or folder send a DELETE request to the delete link. Nothing will be returned in the response body. As a precaution against inadvertantly deleting the root folder, the query parameter `confirm_delete` must be set to `1` for root folder deletes. In addition, a root folder delete does not actually delete the root folder. Instead it deletes all contents of the folder, but not the folder itself.

2.3.5 Magic Query Parameters

Provider Handler Params

These query parameters apply to all providers. These are used, along with the request method, to specify what operation to perform, whether to upload, download, move, rename .etc.

meta

Indicates that WaterButler should return metadata about the file instead of downloading the contents. Not necessary for folders, which return metadata by default.

- **Type**: flag
- **Expected on**: `GET` requests for files
- **Interactions**:
  - `revisions` / `versions`: `meta` takes precedence. File metadata is returned, the revision list is not.
  - `revision` / `version`: These are honored and passed to the the metadata method. Metadata for the file at the specified revision is returned.
- **Notes**:
  - The `meta` query parameter is not required to fetch folder metadata; a bare `GET` folder request suffices. To download a folder, the `zip` query parameter should be provided.

zip

Tells WaterButler to download a folder's contents as a .zip file.

- **Type**: flag
- **Expected on**: `GET` requests against folder paths
- **Interactions**:
  - Take precendence over all other query parameters, which will be ignored.
- **Notes**:
  - A `GET` request against a folder with no query parameters will return metadata, but the same request on a file will download it.

kind

Indicates whether a `PUT` request should create a file or a folder.

- **Type**: string, either "`file`" or "`folder`", defaulting to "`file`"
- **Expected on**: `PUT` requests
- **Interactions**: None
- **Notes**:
  - Issuing a `PUT` request against a file with `?kind=folder` will always fail, throwing a `400 Bad Request`.

name

Indicates the name of the file or folder to be created.

- **Type**: string
- **Expected on**: `PUT` requests for folders

- **Interactions**: None
- **Notes**:
  - The `name` parameter is only valid when creating a new file or folder. Including it in a `PUT` request against a file will result in a `400 Bad Request`. Renaming files is done with `POST` requests.

revisions / versions

Indicates the user wants a list of metadata for all available file revisions.

- **Type**: flag
- **Expected on**: `GET` for file paths
- **Interactions**:
  - Both parameters are overridden by the `meta` parameter. Neither should be used with other parameters.
  - `revisions` and `versions` are currently used interchangeably, with `versions` taking precedence if both are provided.
- **Notes**:
  - The pluralization is vital, `version` and `revision` are used for identifying particular versions.

revision / version

This is the id of the version or revision of the file or folder which Waterbuter is to return.

- **Type**: int
- **Expected on**: `GET` or `HEAD` requests for files or folders
- **Interactions**:
  - is used as a parameter of the metadata provider function.
- **Notes**:
  - If both are provided, `version` takes precendence over `revision`.
  - `revision` and `version` can be used interchangeably. Comments within the code indicate `version` is preferred, but no reason is supplied.
  - Note the lack of pluralization.

direct

Issuing a download request with a query parameter named `direct` indicates that WB should handle the download, even if a direct download via redirect would be possible (e.g. osfstorage and s3). In this case, WB will act as a middleman, downloading the data from the provider and passing it through to the requestor.

- **Type**: flag
- **Expected on**: `GET` file paths
- **Interactions**: None
- **Notes**:
  - Only supported by/relevant to osfstorage (GoogleCloud or Rackspace Cloudfiles backend) and S3.

displayName

When downloading a file, sets the name to download it as. Replaces the original file name in the Content-Disposition header.

- **Type**: string
- **Expected on**: `GET` download requests for files
- **Interactions**: None
- **Notes**: None

mode

Indicates if a file is being downloaded to be rendered. Outside OSF's MFR this isn't useful.

- **Type**: string
- **Expected on**: `GET` requests for files
- **Interactions**: None
- **Notes**:
  - `mode` is only used by the osfstorage provider for MFR.

confirm_delete

WaterButler does not permit users to delete the root folder of a provider, as this would break the connection between the resource and the storage provider. This request has been repurposed to recursively delete the entire contents of the root, leaving the root behind. For safety, this request requires an additional query parameter `confirm_delete` to be present and set to `1`.

- **Type**: bool
- **Expected on**: `DELETE` requests against a root folder
- **Interactions**: None
- **Notes**:
  - Currently supported by: Figshare, Dropbox, Box, Github, S3, Google Drive, and osfstorage

Auth Handler Params

These query parameters are relayed to the auth handler to support authentication and authorization of the request.

cookie

Allows WaterButler to authenticate as a user using a cookie issued by the auth handler.

- **Type**: string
- **Expected on**: All calls
- **Notes**: This is a legacy method of authentication and will be discontinued in the future.

view_only

OSF-specific parameter used to identify special "view-only" links that are used to give temporary read access toa protected resource.

- **Type**: string
- **Expected on**: `GET` requests for files or folders
- **Notes**: Only used internally for the Open Science Framework.

GitHub Provider Params

Query parameters specific to the GitHub provider.

commit / branch identification

Not a single parameter, but rather a class of parameters. WaterButler has used many different parameters to identify the branch or commit a particular file should be found under. These parameters can be either a commit SHA or a branch name. These parameters are `ref`, `version`, `branch`, `sha`, `revision`. All will continue to be supported to maintain back-compatibility, but `ref` (for SHAs or branch names) and `branch` (branch names only) are preferred.

If both a SHA and a branch name are provided in different parameters, the SHA will take precedence. If multiple parameters are given with different SHAs, then the order of precedence will be: `ref`, `version`, `sha`, `revision`, `branch`. If multiple parameters are given with different branches, the order of precedence is: `branch`, `ref`, `version`, `sha`, `revision`.

- **Type**: str
- **Expected on**: Any GitHub provider request
- **Interactions**: None

fileSha

Identifies a specific revision of a file via its SHA.

> **Warning: PLEASE DON'T USE THIS!** This was a mistake and should have never been added. It will hopefully be removed or at the very least demoted in a future version. File SHAs only identify the contents of a file. They provide no information about the file name, path, commit, branch, etc.

- **Type**: str
- **Expected on**: Any GitHub provider request
- **Interactions**: The `fileSha` is always assumed to be a file revision that is an ancestor of the imputed commit or branch ref. Providing a `fileSha` for a file version that was committed after the imputed ref will result in a 404.

2.4 Providers

2.4.1 Base Provider

2.4.2 Amazon S3 Provider

Metadata

Settings

2.4.3 Box Provider

2.4.4 Dataverse Provider

Metadata

Settings

2.4.5 Dropbox Provider

2.4.6 Figshare Provider

2.4.7 File System Provider

> **Warning:** This Provider is for debugging purposes only do **not** use it in production.

2.4.8 Github Provider

2.4.9 GitLab Provider

2.4.10 Google Cloud Provider

Metadata

Utils

2.4.11 Google Drive Provider

2.4.12 OneDrive Provider

2.4.13 OSFStorage Provider

Metadata

2.4.14 ownCloud Storage Provider

2.4.15 Rackspace CloudFiles Provider

2.5 Adding A New Provider

The job of the provider is to translate our common RESTful API into actions against the external provider. The WaterButler API v1 handler (waterbutler.server.api.v1.provider) accepts the incoming requests, builds the appropriate provider object, does some basic validation on the inputs, then passes the request data off to the provider action method. A new provider will inherit from `waterbutler.core.provider.BaseProvider` and implement some or all of the following methods:

```
validate_path()          abstract
validate_v1_path()       abstract
download()               abstract
metadata()               abstract
upload()                 abstract
delete()                 abstract
can_duplicate_names()    abstract
create_folder()          error (405 Not Supported)
intra_copy()             error (501 Not Implemented)
intra_move()             default
can_intra_copy()         default
can_intra_move()         default
exists()                 default
revalidate_path()        default
zip()                    default
path_from_metadata()     default
revisions()              default
shares_storage_root()    default
move()                   default
copy()                   default
handle_naming()          default
handle_name_conflict()   default
```

The methods labeled `abstract` must be implemented. The methods labeled `error` do not need to be implemented, but will raise errors if a user accesses them. The methods labeled `default` have default implementations that may suffice depending on the provider.

2.6 Rate-limiting

As of the v21.2.0 release, WaterButler has built-in rate-limiting via redis. The implementation uses the fixed window algorithm.

2.6.1 Method

Users are distinguished first by their credentials and then by their IP address. The rate limiter recognizes different types of auth and rate-limits each type separately. The four recognized auth types are: OSF cookie, OAuth bearer token, basic auth with base64-encoded username/password, and un-authed.

OSF cookies, OAuth access tokens, and base64-encoded usernames/passwords are used as redis keys during rate-limiting. WB obfuscates them for the same reason that only password hashes are stored in a database. SHA-256 is used in this case. A prefix is also added to the digest to identify which type it is. The "No Auth" case is hashed as well (unnecessarily) so that the keys all have the same look and length.

Auth by OSF cookie currently bypasses the rate limiter to avoid throttling web users.

2.6.2 Configuration

Rate limiting settings are found in *waterbutler.server.settings*. By default, WB allows 3600 requests per auth per hour. Rate-limiting is turned OFF by default; set `ENABLE_RATE_LIMITING` to `True` turn it on. The relevant envvars are:

- `SERVER_CONFIG_ENABLE_RATE_LIMITING`: `Boolean`. Defaults to `False`.
- `SERVER_CONFIG_REDIS_HOST`: The host redis is listening on. Default is `'192.168.168.167'`.
- `SERVER_CONFIG_REDIS_PORT`: The port redis is listening on. Default is `'6379'`.
- `SERVER_CONFIG_REDIS_PASSWORD`: The password for the configured redis instance. Default is `None`.
- `SERVER_CONFIG_RATE_LIMITING_FIXED_WINDOW_SIZE`: Number of seconds until the redis key expires. Default is 3600s.
- `SERVER_CONFIG_RATE_LIMITING_FIXED_WINDOW_LIMIT`: Number of reqests permitted while the redis key is active. Default is 3600.

2.6.3 Behavior

Return the Retry-After header in the 429 response if the limit is hit. This header states when it will be acceptable to send another request. Other informative headers are included to provide context, though currently only after the rate limiting has been enforced.

If rate-limiting is enabled and WB is unable to reach redis, a 503 Service Unavailable error will be thrown. Since redis is not expected to be available during ci, rate limiting is turned off.

2.7 Code

Important code links

2.7.1 waterbutler package

waterbutler.constants module

Constants

waterbutler.settings module

class waterbutler.settings.**SettingsDict**(**args*, *parent=None*, ***kwargs*)

Bases: `dict`

Allow overriding on-disk config via environment variables. Normal config is done with a hierarchical dict:

```
"SERVER_CONFIG": {
  "HOST": "http://localhost:7777"
}
```

`HOST` can be retrieved in the python code with:

```
config = SettingsDict(json.load('local-config.json'))
server_cfg = config.child('SERVER_CONFIG')
host = server_cfg.get('HOST')
```

To override a value, join all of the parent keys and the child keys with an underscore:

```
$ SERVER_CONFIG_HOST='http://foo.bar.com' invoke server
```

Nested dicts can be handled with the `.child()` method. Config keys will be all parent keys joined by underscores:

```
"SERVER_CONFIG": {
  "ANALYTICS": {
    "PROJECT_ID": "foo"
  }
}
```

The corresponding envvar for `PROJECT_ID` would be `SERVER_CONFIG_ANALYTICS_PROJECT_ID`.

get(*key*, *default=None*)
Fetch a config value for `key` from the settings. First checks the env, then the on-disk config. If neither exists, returns `default`.

get_bool(*key*, *default=None*)
Fetch a config value and interpret as a bool. Since envvars are always strings, interpret '0' and the empty string as False and '1' as True. Anything else is probably an acceident, so die screaming.

get_nullable(*key*, *default=None*)
Fetch a config value and interpret the empty string as None. Useful for external code that expects an explicit None.

get_object(*key*, *default=None*)
Fetch a config value and interpret as a Python object or list. Since envvars are always strings, interpret values of type `str` as JSON object or array. Otherwise assume the type is already a python object.

full_key(*key*)
The name of the envvar which corresponds to this key.

child(*key*)
Fetch a sub-dict of the current dict.

`waterbutler.settings.`**child**(*key*)

waterbutler.sizes module

A utility module for writing legible static numbers >>> 10 * MBs >>> 6 * GBs

waterbutler.version module

Module contents

2.7.2 waterbutler.core package

waterbutler.core.auth module

class waterbutler.core.auth.**AuthType**
Bases: enum.Enum

An enumeration.

SOURCE = 0

DESTINATION = 1

class waterbutler.core.auth.**BaseAuthHandler**
Bases: object

fetch(*request*, *bundle*)

get(*resource*, *provider*, *request*, *action=None*, *auth_type=<AuthType.SOURCE: 0>*, *path=''*, *version=None*)

waterbutler.core.exceptions module

waterbutler.core.log_payload module

waterbutler.core.logging module

class waterbutler.core.logging.**MaskFormatter**(*fmt=None*, *datefmt=None*, *style='%'*, *pattern=None*, *mask='***'*)
Bases: logging.Formatter

format(*record*)
Format the specified record as text.

The record's attribute dictionary is used as the operand to a string formatting operation which yields the returned string. Before formatting the dictionary, a couple of preparatory steps are carried out. The message attribute of the record is computed using LogRecord.getMessage(). If the formatting string uses the time (as determined by a call to usesTime(), formatTime() is called to format the event time. If there is exception information, it is formatted using formatException() and appended to the message.

waterbutler.core.metadata module

waterbutler.core.metrics module

class waterbutler.core.metrics.**MetricsBase**
Bases: object

Lightweight wrapper around a dict to make keeping track of metrics a little easier.

Current functionality is limited, but may be extended later. To do:

- update/override method to indicate expectations of existing key

  self.metrics.add_default('some.flag', True) <later> self.metrics.override('some.flag', False) # dies if 'some.flag' doesn't already exist

- optional type validation?

self.metrics.add('some.flag', True, bool()) -or- self.metrics.define('some.flag', bool()) <later> self.metrics.add('some.flag', 'foobar') # dies, 'foobar' isn't a bool

key()
ID string for this object

add(*key*, *value*)
add() stores the given value under the given key. Subkeys can be specified by placing a dot between the parent and child keys. e.g. 'foo.bar' will be interpreted as `self._metrics['foo']['bar']`

Parameters

- **key** (*str*) – the key to store `value` under
- **value** – the value to store, type unrestricted

incr(*key*)
incr() increments the value stored in key, or initializes it to 1 if it has not yet been set.

Parameters **key** (*str*) – the key to increment the `value` of

append(*key*, *new_value*)
Assume key points to a list and append `new_value` to it. Will initialize a list if key is undefined. Type homogeneity of list members is not enforced.

Parameters

- **key** (*str*) – the key to store `value` under
- **value** – the value to store, type unrestricted

merge(*record*)
Merges a dict into the current metrics.

Parameters **record** (*dict*) – a dict to merge with the current metrics

serialize()
Return a copy of the metrics

manifesto()
'This is who I am and this is what I stand for!'

Returns a dict with one entry: our key pointing to our metrics

class waterbutler.core.metrics.**MetricsRecord**(*category*)
Bases: *waterbutler.core.metrics.MetricsBase*

An extension to MetricsBase that carries a category and list of submetrics. When serialized, will include the serialized child metrics

key
ID string for this record: '{category}'

serialize()
Returns its metrics with the metrics for each of the subrecords included under their key.

new_subrecord(*name*)
Create a new MetricsSubRecord object with our category and save it to the subrecords list.

class waterbutler.core.metrics.**MetricsSubRecord**(*category*, *name*)
Bases: *waterbutler.core.metrics.MetricsRecord*

An extension to MetricsRecord that carries a name in addition to a category. Will identify itself as {category}_{name}. Can create its own subrecord whose category will be this subrecord's `name`.

key
ID string for this subrecord: '{category}_{name}'

new_subrecord(*name*)
Creates and saves a new subrecord. The new subrecord will have its category set to the parent subrecord's `name`. ex:

```
parent = MetricsRecord('foo')
child = parent.new_subrecord('bar')
grandchild = child.new_subrecord('baz')

print(parent.key)       # foo
print(child.key)        # foo_bar
print(grandchild.key)   # bar_baz
```

waterbutler.core.path module

waterbutler.core.provider module

waterbutler.core.remote_logging module

waterbutler.core.signing module

waterbutler.core.signing.**order_recursive**(*data*)
Recursively sort keys of input data and all its nested dictionaries. Used to ensure consistent ordering of JSON payloads.

waterbutler.core.signing.**serialize_payload**(*payload*)

waterbutler.core.signing.**unserialize_payload**(*message*)

class waterbutler.core.signing.**Signer**(*secret*, *digest*)
Bases: `object`

sign_message(*message*)

sign_payload(*payload*)

verify_message(*signature*, *message*)

verify_payload(*signature*, *payload*)

waterbutler.core.signing.**sign_data**(*signer*, *data*, *ttl=100*)

waterbutler.core.utils module

Module contents

2.7.3 waterbutler.core.streams package

BaseStream

class waterbutler.core.streams.**BaseStream**(**args*, ***kwargs*)
A wrapper class around an existing stream that supports teeing to multiple reader and writer objects. Though it inherits from `asyncio.StreamReader` it does not implement/augment all of its methods. Only `read()` implements the teeing behavior; `readexactly`, `readline`, and `readuntil` do not.

Classes that inherit from *BaseStream* must implement a `_read()` method that reads `size` bytes from its source and returns it.

size

add_reader(*name*, *reader*)

remove_reader(*name*)

add_writer(*name*, *writer*)

remove_writer(*name*)

feed_eof()

read(*size=-1*)

Read up to `n` bytes from the stream.

If n is not provided, or set to -1, read until EOF and return all read bytes. If the EOF was received and the internal buffer is empty, return an empty bytes object.

If n is zero, return empty bytes object immediately.

If n is positive, this function try to read `n` bytes, and may return less or equal bytes than requested, but at least one byte. If EOF was received before any byte is read, this function returns empty byte object.

Returned value is not limited with limit, configured at stream creation.

If stream was paused, this function will automatically resume it if needed.

ResponseStreamReader

class waterbutler.core.streams.**ResponseStreamReader**(*response*, *size=None*, *name=None*)

Bases: `waterbutler.core.streams.base.BaseStream`

partial

content_type

content_range

name

size

add_reader(*name*, *reader*)

add_writer(*name*, *writer*)

at_eof()

Return True if the buffer is empty and 'feed_eof' was called.

exception()

feed_data(*data*)

feed_eof()

read(*size=-1*)

Read up to `n` bytes from the stream.

If n is not provided, or set to -1, read until EOF and return all read bytes. If the EOF was received and the internal buffer is empty, return an empty bytes object.

If n is zero, return empty bytes object immediately.

If n is positive, this function try to read `n` bytes, and may return less or equal bytes than requested, but at least one byte. If EOF was received before any byte is read, this function returns empty byte object.

Returned value is not limited with limit, configured at stream creation.

If stream was paused, this function will automatically resume it if needed.

readexactly(*n*)
Read exactly `n` bytes.

Raise an IncompleteReadError if EOF is reached before `n` bytes can be read. The IncompleteReadError.partial attribute of the exception will contain the partial read bytes.

if n is zero, return empty bytes object.

Returned value is not limited with limit, configured at stream creation.

If stream was paused, this function will automatically resume it if needed.

readline()
Read chunk of data from the stream until newline (b' ') is found.

On success, return chunk that ends with newline. If only partial line can be read due to EOF, return incomplete line without terminating newline. When EOF was reached while no bytes read, empty bytes object is returned.

If limit is reached, ValueError will be raised. In that case, if newline was found, complete line including newline will be removed from internal buffer. Else, internal buffer will be cleared. Limit is compared against part of the line without newline.

If stream was paused, this function will automatically resume it if needed.

readuntil(*separator=b'\n'*)
Read data from the stream until `separator` is found.

On success, the data and separator will be removed from the internal buffer (consumed). Returned data will include the separator at the end.

Configured stream limit is used to check result. Limit sets the maximal length of data that can be returned, not counting the separator.

If an EOF occurs and the complete separator is still not found, an IncompleteReadError exception will be raised, and the internal buffer will be reset. The IncompleteReadError.partial attribute may contain the separator partially.

If the data cannot be read because of over limit, a LimitOverrunError exception will be raised, and the data will be left in the internal buffer, so it can be read again.

remove_reader(*name*)

remove_writer(*name*)

set_exception(*exc*)

set_transport(*transport*)

RequestStreamReader

class waterbutler.core.streams.**RequestStreamReader**(*request*, *inner*)
Bases: `waterbutler.core.streams.base.BaseStream`

size

add_reader(*name*, *reader*)

add_writer(*name*, *writer*)

at_eof()
Return True if the buffer is empty and 'feed_eof' was called.

exception()

feed_data(*data*)

feed_eof()

read(*size=-1*)
Read up to `n` bytes from the stream.

If n is not provided, or set to -1, read until EOF and return all read bytes. If the EOF was received and the internal buffer is empty, return an empty bytes object.

If n is zero, return empty bytes object immediately.

If n is positive, this function try to read `n` bytes, and may return less or equal bytes than requested, but at least one byte. If EOF was received before any byte is read, this function returns empty byte object.

Returned value is not limited with limit, configured at stream creation.

If stream was paused, this function will automatically resume it if needed.

readexactly(*n*)
Read exactly `n` bytes.

Raise an IncompleteReadError if EOF is reached before `n` bytes can be read. The IncompleteReadError.partial attribute of the exception will contain the partial read bytes.

if n is zero, return empty bytes object.

Returned value is not limited with limit, configured at stream creation.

If stream was paused, this function will automatically resume it if needed.

readline()
Read chunk of data from the stream until newline (b' ') is found.

On success, return chunk that ends with newline. If only partial line can be read due to EOF, return incomplete line without terminating newline. When EOF was reached while no bytes read, empty bytes object is returned.

If limit is reached, ValueError will be raised. In that case, if newline was found, complete line including newline will be removed from internal buffer. Else, internal buffer will be cleared. Limit is compared against part of the line without newline.

If stream was paused, this function will automatically resume it if needed.

readuntil(*separator=b'\n'*)
Read data from the stream until `separator` is found.

On success, the data and separator will be removed from the internal buffer (consumed). Returned data will include the separator at the end.

Configured stream limit is used to check result. Limit sets the maximal length of data that can be returned, not counting the separator.

If an EOF occurs and the complete separator is still not found, an IncompleteReadError exception will be raised, and the internal buffer will be reset. The IncompleteReadError.partial attribute may contain the separator partially.

If the data cannot be read because of over limit, a LimitOverrunError exception will be raised, and the data will be left in the internal buffer, so it can be read again.

remove_reader(*name*)

remove_writer(*name*)

set_exception(*exc*)

set_transport(*transport*)

FileStreamReader

class waterbutler.core.streams.**FileStreamReader**(*file_pointer*)
Bases: waterbutler.core.streams.base.BaseStream

size

close()

add_reader(*name*, *reader*)

add_writer(*name*, *writer*)

at_eof()
Return True if the buffer is empty and 'feed_eof' was called.

chunk_reader()

exception()

feed_data(*data*)

feed_eof()

read(*size=-1*)
Read up to n bytes from the stream.

If n is not provided, or set to -1, read until EOF and return all read bytes. If the EOF was received and the internal buffer is empty, return an empty bytes object.

If n is zero, return empty bytes object immediately.

If n is positive, this function try to read n bytes, and may return less or equal bytes than requested, but at least one byte. If EOF was received before any byte is read, this function returns empty byte object.

Returned value is not limited with limit, configured at stream creation.

If stream was paused, this function will automatically resume it if needed.

readexactly(*n*)
Read exactly n bytes.

Raise an IncompleteReadError if EOF is reached before n bytes can be read. The IncompleteReadError.partial attribute of the exception will contain the partial read bytes.

if n is zero, return empty bytes object.

Returned value is not limited with limit, configured at stream creation.

If stream was paused, this function will automatically resume it if needed.

readline()
Read chunk of data from the stream until newline (b' ') is found.

On success, return chunk that ends with newline. If only partial line can be read due to EOF, return incomplete line without terminating newline. When EOF was reached while no bytes read, empty bytes object is returned.

If limit is reached, ValueError will be raised. In that case, if newline was found, complete line including newline will be removed from internal buffer. Else, internal buffer will be cleared. Limit is compared against part of the line without newline.

If stream was paused, this function will automatically resume it if needed.

readuntil(*separator=b'\n'*)

Read data from the stream until `separator` is found.

On success, the data and separator will be removed from the internal buffer (consumed). Returned data will include the separator at the end.

Configured stream limit is used to check result. Limit sets the maximal length of data that can be returned, not counting the separator.

If an EOF occurs and the complete separator is still not found, an IncompleteReadError exception will be raised, and the internal buffer will be reset. The IncompleteReadError.partial attribute may contain the separator partially.

If the data cannot be read because of over limit, a LimitOverrunError exception will be raised, and the data will be left in the internal buffer, so it can be read again.

remove_reader(*name*)

remove_writer(*name*)

set_exception(*exc*)

set_transport(*transport*)

HashStreamWriter

class waterbutler.core.streams.**HashStreamWriter**(*hasher*)

Bases: `object`

Stream-like object that hashes and discards its input.

digest

hexdigest

can_write_eof()

write(*data*)

close()

StringStream

class waterbutler.core.streams.**StringStream**(*data*)

Bases: `waterbutler.core.streams.base.BaseStream`

size

add_reader(*name*, *reader*)

add_writer(*name*, *writer*)

at_eof()

Return True if the buffer is empty and 'feed_eof' was called.

exception()

feed_data(*data*)

feed_eof()

read(*size=-1*)
Read up to n bytes from the stream.

If n is not provided, or set to -1, read until EOF and return all read bytes. If the EOF was received and the internal buffer is empty, return an empty bytes object.

If n is zero, return empty bytes object immediately.

If n is positive, this function try to read n bytes, and may return less or equal bytes than requested, but at least one byte. If EOF was received before any byte is read, this function returns empty byte object.

Returned value is not limited with limit, configured at stream creation.

If stream was paused, this function will automatically resume it if needed.

readexactly(*n*)
Read exactly n bytes.

Raise an IncompleteReadError if EOF is reached before n bytes can be read. The IncompleteReadError.partial attribute of the exception will contain the partial read bytes.

if n is zero, return empty bytes object.

Returned value is not limited with limit, configured at stream creation.

If stream was paused, this function will automatically resume it if needed.

readline()
Read chunk of data from the stream until newline (b' ') is found.

On success, return chunk that ends with newline. If only partial line can be read due to EOF, return incomplete line without terminating newline. When EOF was reached while no bytes read, empty bytes object is returned.

If limit is reached, ValueError will be raised. In that case, if newline was found, complete line including newline will be removed from internal buffer. Else, internal buffer will be cleared. Limit is compared against part of the line without newline.

If stream was paused, this function will automatically resume it if needed.

readuntil(*separator=b'\n'*)
Read data from the stream until separator is found.

On success, the data and separator will be removed from the internal buffer (consumed). Returned data will include the separator at the end.

Configured stream limit is used to check result. Limit sets the maximal length of data that can be returned, not counting the separator.

If an EOF occurs and the complete separator is still not found, an IncompleteReadError exception will be raised, and the internal buffer will be reset. The IncompleteReadError.partial attribute may contain the separator partially.

If the data cannot be read because of over limit, a LimitOverrunError exception will be raised, and the data will be left in the internal buffer, so it can be read again.

remove_reader(*name*)

remove_writer(*name*)

set_exception(*exc*)

set_transport(*transport*)

MultiStream

class waterbutler.core.streams.**MultiStream**(**streams*)
Bases: asyncio.streams.StreamReader

Concatenate a series of StreamReader objects into a single stream. Reads from the current stream until exhausted, then continues to the next, etc. Used to build streaming form data for Figshare uploads. Originally written by @jmcarp

size

streams

add_streams(**streams*)

at_eof()
Return True if the buffer is empty and 'feed_eof' was called.

exception()

feed_data(*data*)

feed_eof()

read(*n=-1*)
Read up to n bytes from the stream.

If n is not provided, or set to -1, read until EOF and return all read bytes. If the EOF was received and the internal buffer is empty, return an empty bytes object.

If n is zero, return empty bytes object immediately.

If n is positive, this function try to read n bytes, and may return less or equal bytes than requested, but at least one byte. If EOF was received before any byte is read, this function returns empty byte object.

Returned value is not limited with limit, configured at stream creation.

If stream was paused, this function will automatically resume it if needed.

readexactly(*n*)
Read exactly n bytes.

Raise an IncompleteReadError if EOF is reached before n bytes can be read. The IncompleteReadError.partial attribute of the exception will contain the partial read bytes.

if n is zero, return empty bytes object.

Returned value is not limited with limit, configured at stream creation.

If stream was paused, this function will automatically resume it if needed.

readline()
Read chunk of data from the stream until newline (b' ') is found.

On success, return chunk that ends with newline. If only partial line can be read due to EOF, return incomplete line without terminating newline. When EOF was reached while no bytes read, empty bytes object is returned.

If limit is reached, ValueError will be raised. In that case, if newline was found, complete line including newline will be removed from internal buffer. Else, internal buffer will be cleared. Limit is compared against part of the line without newline.

If stream was paused, this function will automatically resume it if needed.

readuntil(*separator=b'\n'*)
Read data from the stream until `separator` is found.

On success, the data and separator will be removed from the internal buffer (consumed). Returned data will include the separator at the end.

Configured stream limit is used to check result. Limit sets the maximal length of data that can be returned, not counting the separator.

If an EOF occurs and the complete separator is still not found, an IncompleteReadError exception will be raised, and the internal buffer will be reset. The IncompleteReadError.partial attribute may contain the separator partially.

If the data cannot be read because of over limit, a LimitOverrunError exception will be raised, and the data will be left in the internal buffer, so it can be read again.

set_exception(*exc*)

set_transport(*transport*)

FormDataStream

class waterbutler.core.streams.**FormDataStream**(***fields*)
Bases: `waterbutler.core.streams.base.MultiStream`

A child of MultiSteam used to create stream friendly multipart form data requests. Usage:

```
>>> stream = FormDataStream(key1='value1', file=FileStream(...))
```

Or:

```
>>> stream = FormDataStream()
>>> stream.add_field('key1', 'value1')
>>> stream.add_file('file', FileStream(...), mime='text/plain')
```

Additional options for files can be passed as a tuple ordered as:

```
>>> FormDataStream(fieldName=(FileStream(...), 'fileName', 'Mime', 'encoding'))
```

Auto generates boundaries and properly concatenates them Use FormDataStream.headers to get the proper headers to be included with requests Namely Content-Length, Content-Type

Parameters **fields** (*dict*) – A dict of fieldname: value to create the body of the stream

classmethod make_boundary()
Creates a random-ish boundary for form data separator

classmethod make_header(*name*, *disposition='form-data'*, *additional_headers=None*, ***extra*)

end_boundary

headers
The headers required to make a proper multipart form request Implicitly calls finalize as accessing headers will often indicate sending of the request Meaning nothing else will be added to the stream

finalize()

add_fields(***fields*)

add_field(*key*, *value*)

add_file(*field_name*, *file_stream*, *file_name=None*, *mime='application/octet-stream'*, *disposition='file'*, *transcoding='binary'*)

add_streams(**streams*)

at_eof()
Return True if the buffer is empty and 'feed_eof' was called.

exception()

feed_data(*data*)

feed_eof()

read(*n=-1*)
Read up to `n` bytes from the stream.

If n is not provided, or set to -1, read until EOF and return all read bytes. If the EOF was received and the internal buffer is empty, return an empty bytes object.

If n is zero, return empty bytes object immediately.

If n is positive, this function try to read `n` bytes, and may return less or equal bytes than requested, but at least one byte. If EOF was received before any byte is read, this function returns empty byte object.

Returned value is not limited with limit, configured at stream creation.

If stream was paused, this function will automatically resume it if needed.

readexactly(*n*)
Read exactly `n` bytes.

Raise an IncompleteReadError if EOF is reached before `n` bytes can be read. The IncompleteReadError.partial attribute of the exception will contain the partial read bytes.

if n is zero, return empty bytes object.

Returned value is not limited with limit, configured at stream creation.

If stream was paused, this function will automatically resume it if needed.

readline()
Read chunk of data from the stream until newline (b' ') is found.

On success, return chunk that ends with newline. If only partial line can be read due to EOF, return incomplete line without terminating newline. When EOF was reached while no bytes read, empty bytes object is returned.

If limit is reached, ValueError will be raised. In that case, if newline was found, complete line including newline will be removed from internal buffer. Else, internal buffer will be cleared. Limit is compared against part of the line without newline.

If stream was paused, this function will automatically resume it if needed.

readuntil(*separator=b'\n'*)
Read data from the stream until `separator` is found.

On success, the data and separator will be removed from the internal buffer (consumed). Returned data will include the separator at the end.

Configured stream limit is used to check result. Limit sets the maximal length of data that can be returned, not counting the separator.

If an EOF occurs and the complete separator is still not found, an IncompleteReadError exception will be raised, and the internal buffer will be reset. The IncompleteReadError.partial attribute may contain the separator partially.

If the data cannot be read because of over limit, a LimitOverrunError exception will be raised, and the data will be left in the internal buffer, so it can be read again.

set_exception(*exc*)

set_transport(*transport*)

size

streams

CutoffStream

class waterbutler.core.streams.**CutoffStream**(*stream*, *cutoff*)

A wrapper around an existing stream that terminates after pulling off the specified number of bytes. Useful for segmenting an existing stream into parts suitable for chunked upload interfaces.

This class only subclasses `asyncio.StreamReader` to take advantage of the `isinstance`-based stream-reading interface of aiohttp v0.18.2. It implements a `read()` method with the same signature as `StreamReader` that does the bookkeeping to know how many bytes to request from the stream attribute.

Parameters

- **stream** – a stream object to wrap
- **cutoff** (*int*) – number of bytes to read before stopping

size

The lesser of the wrapped stream's size or the cutoff.

read(*n=-1*)

Read `n` bytes from the stream. `n` is a chunk size, not the full size of the stream. If `n` is -1, read `cutoff` bytes. If `n` is a positive integer, read that many bytes as long as the total number of bytes read so far does not exceed `cutoff`.

2.7.4 waterbutler.providers package

Subpackages

Module contents

2.7.5 waterbutler.server package

waterbutler.server.app module

waterbutler.server.settings module

waterbutler.server.utils module

waterbutler.server.utils.**parse_request_range**(*range_header*)

WB uses tornado's `httputil._parse_request_range` function to parse the Range HTTP header and return a tuple representing the range. Tornado's version returns a tuple suitable for slicing arrays, meaning that a range of 0-1 will be returned as `(0, 2)`. WB had been assuming that the tuple would represent the first and last byte positions and was consistently returning one more byte than requested. Since WB doesn't ever use ranges to do list slicing of byte streams, this function wraps tornado's version and returns the actual byte indices.

Ex. `Range: bytes=0-1` will be returned as `(0, 1)`.

If the end byte is omitted, the second element of the tuple will be `None`. This will be sent to the provider as an open ended range, e.g. (`Range: bytes=5-`). Most providers interpret this to mean "send from the start byte to the end of the file".

If this function receives an unsupported or unfamiliar Range header, it will return `None`, indicating that the full file should be sent. Some formats supported by other providers but unsupported by WB include:

- `Range: bytes=-5` – some providers interpret this as "send the last five bytes"
- `Range: bytes=0-5,10-12` – indicates a multi-range, "send the first six bytes, then the next three bytes starting from the eleventh".

Unfamiliar byte ranges are anything not matching `^bytes=[0-9]+\-[0-9]*$`, or ranges where the end byte position is less than the start byte.

Parameters **range_header** (*str*) – a string containing the value of the Range header

Return type `tuple` or `None`

Returns a `tuple` representing the inclusive range of byte positions or `None`.

class waterbutler.server.utils.**CORsMixin**

Bases: `object`

set_default_headers()

options(**args*, ***kwargs*)

class waterbutler.server.utils.**UtilMixin**

Bases: `object`

bytes_downloaded = 0

bytes_uploaded = 0

set_status(*code*, *reason=None*)

write_stream(*stream*)

Module contents

2.7.6 waterbutler.tasks package

waterbutler.tasks.app module

waterbutler.tasks.settings module

Module contents

2.8 Releases

See the CHANGELOG.

2.9 Testing

v1 API testing with Postman

Postman is a popular tool for testing APIs. Postman collections and environments are available in `tests/postman` for testing the functionality of the v1 API. This is particularly useful for those updating old, or creating new WaterButler providers. Follow the link for instructions on installing the Postman App or its commandline counterpart Newman.

Quickstart for Newman:

```
npm install -g newman  newman run tests/postman/collections/copy_files.json -e
copy_file_sample.json
```

Specific collection instructions

copy_files:
copy_folders:
crud_cases

copy_files, crud_cases and copy_folders can share the same setup and environment.
crud_cases only requires the first PID to be valid, and does not use the second one.

Setup:

1. Create two projects in OSF. Take note of their IDs. You will need the IDs for the Postman environment file.
2. Setup the provider you wish to test, in each of the two OSF projects. Provider root must be the same in both OSF projects.
3. (Optional) If you wish to use a provider, other then osfstorage, for testing inter provider copies, setup an alternative provider in each of the two OSF projects

Environment file:

1. Make a copy of `tests/postman/environments/copy_file_sample.json` and edit as follows.
2. Update `PID` and `PID2` values with your two project IDs.
3. Update `provider` value with the name of the provider you wish to test.
4. Update `alt_provider` value with the name of the provider you will use for inter provider copy testing.
5. Update `basic_auth` value with the basic auth token representing your login to OSF. This can be found using the Postman App. Open a new request, click on authorization tab, select Basic Auth in Type dropdown. Enter your login and password. Click Update Request. Click on Headers Tab. Take note of the value of Authorization header. The value you are looking for is the rest of the string after "Basic ".
6. `protocol`, `host` and `port` can be left as is assuming you have set up your dev environment in the default manner.

Testing:

1. Import the collection you would like to run from `tests/postman/collections` and the environment file you just updated into the Postman App.
2. **Note:** Importing your environment may give a few errors. It is most likely fine and should still run.
3. Run the imported collections using the imported environment.
4. **Note:** A failed run may leave files and/or folders behind. You will need to manually remove these before starting another run.

CHAPTER 3

Project info

3.1 Contributing

```
# Contributing
Waterbutler uses [semantic versioning](http://semver.org/) `<major>.<minor>.<patch>`
* Patches are reserved for hotfixes only
* Minor versions are for **adding** new functionality or fields
* Minor versions **will not** contain breaking changes to the existing API
    - Any changes **must** be backwards compatible
* Major versions **may** contain breaking changes to the existing API
    - Ideally REST endpoints will be versioned, ie `/.../v<major>/...`


Waterbutler conforms to [the git flow work flow](http://nvie.com/posts/a-successful-
↪git-branching-model/)
In brief this means
* Feature branches should be branched off of develop or a release branch
    - Before submitting a pull request re-merge the source branch
    - **Do not** merge develop if you are working of a release branch and visa versa
* Hotfixes are to be branched off master
    - Hotfix PR should be names hotfix/brief-description
        * Use `-`'s for spaces not `_`'s
        * A hotfix for an issue involving figshare metadata when empty lists are␣
↪returned would be`hotfix/figshare-metadata-empty`
    - When hotfixes are merged a new branch will be created bumping the minor version␣
↪ie `hotfix/0.1.3` and the other PR will be merged into it

Waterbutler expects [pretty pull request, clean commit histories and meaningful␣
↪commit messages](http://justinhileman.info/article/changing-history/)
* Make sure to rebase, `git rebase -i <commitsha>`, to remove pointless commits
    - Pointless commits include but are not limited to
        * Fix flake errors
        * Fix typo
        * Fix test
```

(continues on next page)

(continued from previous page)

```
        * etc
* Follow the guide lines for commit message in the above
    - Don't worry about new lines between bullet points


All Waterbutler code **must** pass [flake8 linting](https://www.python.org/dev/peps/
↪pep-0008/)
* Max line is set to 100 characters
* Tests are not linted, but don't be terrible

Imports are should be ordered in pep8 style but ordered by line length

```python
import abc
import asyncio
import itertools
from urllib import parse

import furl
import aiohttp

from waterbutler.core import streams
from waterbutler.core import exceptions

# Not

import abc
import asyncio
import itertools
from urllib import parse

import aiohttp
import furl

from waterbutler.core import exceptions
from waterbutler.core import streams

```

Other general guide lines
* Keep it simple and readable
* Do not use synchronous 3rd party libraries
* If you don't need `**kwargs` don't use it
* Docstrings and comments make everything better
* Avoid single letter variable names outside of comprehensions
* Write tests
```

3.2 ChangeLog

```
*********
ChangeLog
*********

23.0.0 (2023-08-17)
```

(continues on next page)

(continued from previous page)

```
==================

- S3 Improvements: support sub-folder being configured as add-on root

22.0.1 (2022-07-05)
==================
- Fix: Don't call `json_api_serialized` on a list when a folder name conflict is␣
↪detected.

22.0.0 (2022-06-22)
==================
- Fix: Stop double-reporting certain HTTP errors.  Some error paths were resulting in␣
↪two error
bodies being written out by the server.  This could be confusing for free-text errors␣
↪and
unparseable for json-formatted errors.
- Feature: If an upload request is made, but there is already an existing file with␣
↪that name,
include metadata about the target file in the 409 Conflict response.  This will save␣
↪the caller a
metadata request if they wish to overwrite the existing file.
- Fix: Add missing read-write JSON-API links to OneDrive files and folders.  These␣
↪were overlooked
when OneDrive was converted from read-only to read-write.
- Fix: Update the key used to fetch the download url for OneDrive file revisions.
- Fix: Update the OneDrive intra-copy code to stop sending auth to the monitoring␣
↪endpoint and to
expect the 200 OK response that is observed in practice rather than the 303 See Other␣
↪mentioned in
the api documentation.
- Code: Migrate repo CI from TravisCI to GitHub Actions.

21.2.0 (2021-12-06)
==================
- Feature: Give WaterButler the ability to enforce rate-limits.  This requires a␣
↪reachable redis
instance.  See `docs/rate-limiting.rst` for details on setting up and configuring.

21.1.0 (2021-09-30)
==================
- Fix: Stop breaking Google Doc (.gdoc, .gsheet, .gslide) downloads.  Google Doc␣
↪files are not
downloadable in their native format.  Instead they must be exported to an externally-
↪supported
format (e.g. .gdoc => .docx, .gsheet => .xlsx, .gslide => .pptx).  WB had been␣
↪assuming that a file
with an undefined size was a google doc, but recently GDrive has started reporting␣
↪that Google Docs
have a size of 1kb.  WB now explictly checks the mime-type of the requested file to␣
↪determine
whether to download or export.

21.0.1 (2021-09-14)
==================
- Fix: Properly decode OneDrivePaths for small files uploaded under certain provider␣
↪root
configurations.  Some paths were failing to properly decode unicode and url-encoded␣
↪characters.
```

(continues on next page)

(continued from previous page)

```
21.0.0 (2021-08-12)
===================
- Feature: Add read/write support to OneDrive provider, adapted from the previous␣
 ↪work of
Ryan Casey (@caseyrygt) & Oleksandr Melnykov (@alexandr-melnikov-dev-pro).
- Feature: Update OneDrive provider to support both OneDrive Personal and OneDrive␣
 ↪for School or
Business.  Switch the provider to use the MS Graph API and update documentation to␣
 ↪follow.
- Fix: Return a valid empty zip file when downloading a directory with no contents.
- Code: Remove unneeded `async`s from synchronous Box tests.
- Code: Pin pip to v18.1 on TravisCI to fix dependency resolution.

20.1.0 (2020-11-03)
===================
- Feature: Support storage limits on osfstorage.  The OSF now enforces storage quotas␣
 ↪for OSF
projects.  Update WaterButler to check the project's quota status before executing␣
 ↪uploads,
moves, and copies.

20.0.3 (2020-10-21)
===================
- Feature: Add flags to toggle logging of WaterButler events to specific Keen␣
 ↪collections.

20.0.2 (2020-06-02)
===================
- Fix: Bump sentry-sdk dependency from v0.14.0 to v0.14.4 to avoid potential memory␣
 ↪leak.

20.0.1 (2020-03-09)
===================
- Fix: Extend the default timeout for GET, PUT, and POST requests made by aiohttp.  A␣
 ↪five minute
timeout was added to aiohttp between versions v0.18.2 and v3.6.2.  WaterButler needs␣
 ↪to support large
uploads and downloads, so this has been changed to an hour, configurable by the␣
 ↪`AIOHTTP_TIMEOUT`
envvar.  Global request timeouts should be decided by the proxy set up in front of WB.

20.0.0 (2020-02-18)
===================
- Code: Upgrade core library aiohttp from v0.18.2 to v3.6.2.  WaterButler uses␣
 ↪aiohttp to make
requests to the external providers.  Its interface has changed significantly from v0.
 ↪18 and much
of the internal `BaseProvider.make_request` code was refactored/rewritten.  Each␣
 ↪provider has been
updated to use the new `make_request` code.
  - The newest aiohttp favors context managers for performing requests, which does␣
 ↪not mesh well
    with WB's tendency to pass around response objects.  To work around this, the␣
 ↪core provider now
    manually manages session setup and teardown.  Sessions are closed when the object␣
 ↪is
```

(continues on next page)

(continued from previous page)

```
    garbage-collected.
  - WB is moving away from using request context managers in provider code, where␣
↪passing response
    objects is more common.
  - Update WB's stream classes to be compatible with recent asyncio.StreamReader.
- Code: Requirements / dependency upgrades:
  - Upgrade Tornado from v4.3 to v6.0.
  - Add support for Newrelic monitoring, made possible by the upgrade to Tornado v6.0.
  - Update Sentry logging to replace the deprecated `raven` library with `sentry-sdk`.
  - Remove hack for `certifi` library and upgrade to the most recent version.
  - Replace `agent` library with core async generators.
  - Bump testing libraries to newer versions and update testing code to accommodate.
  - Pin some unpinned dependencies in dev-requirements.txt.
- Fix: Many miscellaneous test updates:
  - Fix tests where copy-n-paste had gone awry.
  - Remove `async` from `async def` tests that never `await`-ed anything.
  - Stop saving unused return values into variables.
  - Replace `assert $foo.called_` with `$foo.assert_called_`.
  - Remove no-longer-valid tests.
  - Move common move/copy task-testing code into a conftest.py.
- Fix: Fix invalid pre-decrement of retry count in Box provider.
- Fix: Support Rackspace Cloudfiles as an osfstorage backend.  During the transition␣
↪to the Google
Cloud backend, the bucket-identifying key in the osfstorage configuration changed␣
↪from `container`
to `bucket`.  Re-enable support for `container` to support legacy cloudfiles backends.
- Fix: Support Figshare uploads where the content hashing has not completed before␣
↪Figshare
responds.  Previously, uploads less than 70Mb in size always had the checksum␣
↪embedded in the
response metadata.  Now the availability of the hash appears to be subject to some␣
↪limit on
Figshare's side.  If present, verify that it matches our checksum.  If not, bypass␣
↪the check and
set the `hashingInProgress` metadata field to true.
- Code: Upgrade WB's Dockerfile to be based off python3.6 and Debian buster.  Remove␣
↪manual gosu
installation and fetch from apt-get instead.  python3.6 is now the only supported␣
↪python version.
- Code: Remove old GitLab workarounds.  As a side-effect, the GitLab provider no␣
↪longer supports
`Range` headers for downloads.  To support this would require slurping the file␣
↪contents into
memory (which was done by the workarounds), but this opens WB up to DOS attacks via␣
↪large files.

19.2.0 (2019-10-07)
===================
- Feature: Support rate-limiting in large move/copies for GitHub.  GitHub limits the␣
↪number of
per-user API requests to 5,000 per hour.  To avoid running afoul of these limits,␣
↪slow down the
rate of requests made by WB as we start to approach the rate-limiting threshold. ␣
↪Reserve a
configurable number of requests for regular requests.  If the request limit is␣
↪exhausted by an
external request, WB will throw a `GitHubRateLimitExceededError` error.
```

(continues on next page)

(continued from previous page)

```
- Fix: Consistently handle revision-identifying query parameters in the GitHub␣
↪provider.  Over its
lifetime the GH provider has supported "ref", "sha", "revision", "version", and␣
↪(sometimes)
"branch" as valid revision-identifying parameters.  With this change, all of those␣
↪params are
handled in a single method and in a defined fashion.  Revisions may be commit SHAs or␣
↪branch
names.  A revision is assumed to be a commit SHA if it is a valid 40-digit base-16␣
↪number.
Otherwise it is assumed to be a branch name.  See method documentation for how␣
↪multiple conflicting
parameters are handled.
- Fix: Update figshare provider to report the correct url for publicly-viewable files.
- Fix: Update figshare provider to support "dataset" articles as folders.  figshare␣
↪has many
article types.  Previously, only the "fileset" article type represented a folder in␣
↪the figshare WB
provider. figshare now automatically updates "fileset"s to "dataset"s.  Promote
↪"dataset"s to be the
default folder-analagous article type.

19.1.0 (2019-09-04)
===================
- Fix: Correctly annotate HEAD requests as metadata requests and revision requests as␣
↪revision
requests in the auth payload sent to the OSF.  These were being incorrectly marked as␣
↪download
requests, causing spurious download counts on OSF files.

19.0.1 (2019-08-07)
===================
- Fix: Fix bug in the Box provider that was causing files uploaded to subdirectories␣
↪to end up in
the project root instead.  This would NOT cause an accidental overwrite; if a file␣
↪with the same
name was already present in the project root, Box would throw a 409 Conflict error␣
↪and refuse to
proceed.

19.0.0 (2019-06-11)
===================
- Fix: Bitbucket has retired their v1 API.  Update the WaterButler provider to use v2␣
↪instead.
- Code: Update WaterButler Dockerfile to explicitly use gnupg v2.  The default gpg␣
↪binary on Debian
jessie is v1, but Debian stretch changes this to v2.  Adjust for small differences in␣
↪how they are
called.

18.0.3 (2019-01-31)
===================
- Fix: Send metadata about request to the OSF auth endpoint to help OSF distinguish␣
↪file views from
downloads.  (thanks, @Johnetordoff!)

18.0.2 (2018-12-14)
```

(continues on next page)

(continued from previous page)

```
==================
- Fix: Get CI running again by adding workaround for bad TravisCI/Boto interaction.␣
↪See:
https://github.com/travis-ci/travis-ci/issues/7940.

18.0.1 (2018-12-12)
==================
- Fix: Restore the `filename` parameter to WaterButler's Content-Disposition headers.␣
↪ It had been
removed in favor of `filename*`, which can correctly encode multibyte filenames. ␣
↪This broke scripts
that expected the `filename` version.  Since `filename` cannot support multibyte␣
↪characters, those
characters are converted to their nearest ascii equivalent or stripped if no␣
↪equivalent exists.  WB
now returns both forms of the parameter in the header.  Clients should prefer␣
↪`filename*` for the
most accurate representation of the filename.  (thanks, @jcohenadad!)

18.0.0 (2018-10-31)
==================
- UPDATE: WaterButler is now following the CalVer (https://calver.org/) versioning␣
↪scheme to match
the OSF.  All new versions will use the `YY.MINOR.MICRO` format.
- Fix: Teach WaterButler to properly encode non-ascii file names for download␣
↪requests.  WB was
constructing Content-Disposition headers in many places throughout the codebase. ␣
↪Some correctly
encoded non-ascii characters as UTF-8, but many did not.  These have been␣
↪consolidated into a
single routine that can build explicitly-declared UTF-8 encoded Content-Disposition␣
↪headers.
- Fix: Allow users to set the name of a file when downloading directly from a␣
↪provider.  Some
providers (osfstorage, s3) support signed download urls, which permit the user to␣
↪download directly
from the provider without passing through WB.  WB was failing to relay the␣
↪`displayName` query
parameter to the upstream providers.  It now does so, which allows the user to␣
↪override the default
download file name.  Downloads that are proxied through WB already respect the␣
↪parameter and are
unchanged by this.

0.42.2 (2018-10-24)
==================
- Feature: Relay MFR-identifying header to OSF when requesting auth on MFR's behalf. ␣
↪This enables
more accurate couting of file download and render requests.

0.42.1 (2018-09-17)
==================
- Fix: Fix file updates and overwrites on Dropbox by making sure to set the "overwrite
↪" mode when
replace-on-conflict semantics are requested.

0.42.0 (2018-09-16)
```

(continues on next page)

(continued from previous page)

```
==================
- Feature: Support multi-regional backend storage buckets in osfstorage. WaterButler␣
↪now includes
the path and version of the file when requesting auth from the OSF.  Different␣
↪versions of a file
may be stored in different regions and will therefore require different credentials␣
↪and settings.
This change also requires osfstorage to implement custom move() and copy() methods. ␣
↪Cross-region
osfstorage moves and copies are expected to preserve version history and linked guids.
↪  This is
normally handled by the intra_move() and intra_copy() methods, but those methods only␣
↪operate on
metadata and do not copy data from one region bucket to another. The new move() and␣
↪copy() methods
take care of copying the file data across regions before calling intra_move() and␣
↪intra_copy() to
update the metadata.
- Fix: Stop logging folder metadata requests in v1.  A misunderstanding of the GET␣
↪endpoint for
folders causing some folder metadata requests to be logged as download-as-zip␣
↪requests.
- Code: Remove cold-archiving and parity-generating tasks from osfstorage.  These␣
↪tasks are better
done on the backend storage provider.
- Code: Turn on branch coverage testing. Overall coverage has decreased to 90% with␣
↪this change.

0.41.3 (2018-08-27)
===================
- Feature: Make download-as-zip compression level a tunable configuration parameter.

0.41.2 (2018-08-27)
===================
- Fix: Brown-paper-bag release: Check in settings file needed for last fix.

0.41.1 (2018-08-27)
===================
- Fix: Expand list of file extensions that don't get deflated during download-as-zip.

0.41.0 (2018-08-15)
===================
- Feature: WaterButler now uses the multipart/chunked upload interfaces provided by␣
↪Box, Dropbox,
and Amazon S3 when the uploaded file size exceeds provider-defined thresholds.  These␣
↪providers
limit the size of files that can be uploaded in a single request and require␣
↪multipart uploads for
larger files.  WB itself does not provide a multipart upload interface, so uploads␣
↪via WB will
still appear to happen as a single request. The maximum file sizes for each provider␣
↪at the time
of this release are:
  - Box: 250MB / 2GB / 5GB, depending on account type
    https://community.box.com/t5/Upload-and-Download-Files-and/Understand-the-Maximum-
↪File-Size-You-Can-Upload-to-Box/ta-p/50590
  - Dropbox: 350GB
```

(continues on next page)

(continued from previous page)

```
    https://www.dropbox.com/help/space/upload-limitations
  - S3: 5TB
    https://aws.amazon.com/s3/faqs/ under "How much data can I store in Amazon S3"
- Fix: Improve Figshare upload reliability, especially for cross-provider move/copies.
↪  WaterButler
was trying to slurp an entire file segment into memory before uploading to Figshare. ␣
↪If the source
feed was slow, the connection could time out before the transfer commenced, resulting␣
↪in broken
uploads on Figshare.  WB now streams chunks of the segment as it receives them,␣
↪allowing the
connection to stay open to completion.
- Fix: Fix minor warning in documentation build.

0.40.2 (2018-08-08)
===================
- Fix: Fix download of public Figshare files.

0.40.1 (2018-07-25)
===================
- Feature: Add `X-CSRFToken` to list of acceptable CORS headers.
- Feature: Tell Keen analytics to strip ip on upload.
- Code: Remove never-implemented anonymous geolocation code.

0.40.0 (2018-06-22)
===================
- Feature: Listen for MFR-originating metadata requests and relay the nature of the␣
↪request to
the OSF. This will allow the OSF to tally better metrics on file views.
- Feature: Add new "sizeInt" file metadata field. Some providers return file size as␣
↪a string.
The "size" field in WaterButler's file metadata follows this behavior.  The "sizeInt"␣
↪field will
always be an integer (if file size is available) or `null` (if not).
- Code: Expand tests for WaterButler's APIv1 server.

0.39.2 (2018-06-05)
===================
- Fix: Brown-paper-bag release: actually change version in the code.

0.39.1 (2018-06-05)
===================
- Code: Pin base docker image to python:3.5-slim-jessie.  3.5-slim was recently␣
↪updated to Debian
stretch, and WaterButler has not yet been verified to work on it.

0.39.0 (2018-05-08)
===================
- Feature: WaterButler now lets osfstorage know who the requesting user is when␣
↪listing folder
contents, so that it can return metadata about whether the user has seen the most␣
↪recent version
of the file.  (thanks, @erinspace!)
- Feature: WaterButler includes metadata about the request in the logging callback to␣
↪help the
logger tally more accurate download counts.  (thanks, @johnetordoff!)
- Feature: Stop logging revisions metadata requests to logging callback.  (thanks,␣
↪@johnetordoff!)
```

(continues on next page)

(continued from previous page)

```
- Fix: Don't try to decode the body of HEAD requests that error.  HEAD requests don't␣
↪have bodies!
- Code: Clean up existing mypy annotations and add new ones.  WaterButler still isn't␣
↪100% clean,
but it's getting there!
- Code: Allow passing JSON-encoded objects as config settings via environment␣
↪variables.

0.38.6 (2018-04-25)
===================
- Fix: `CELERY_RESULT_PERSISTENT` should default to True, now that `amqp` is the␣
↪default result
backend.

0.38.5 (2018-04-24)
===================
- Fix: Don't overwrite evokable url with generated url inside the make_request retry␣
↪loop.  Third
time's the charm, right?

0.38.4 (2018-04-24)
===================
- Fix: Evoke url-generating functions *inside* the make_request retry loop to make␣
↪sure signed
urls are as fresh as possible.

0.38.3 (2018-04-24)
===================
- Fix: Remove temporary files after osfstorage provider has completed its archiving␣
↪and
parity-generation tasks.

0.38.2 (2018-04-23)
===================
- Fix: Delay construction of Google Cloud signed urls until right before issuing them.
↪  The provider
was building them outside of the retry loop, resulting in slow-to-fail requests␣
↪having their
signatures expire before all subsequent retries could be issued.

0.38.1 (2018-04-10)
===================
- Fix: Don't log 206 Partial requests to the download callback.  Most of these are␣
↪for streaming
video or resumable downloads and shouldn't be tallied as a download.

0.38.0 (2018-03-28)
===================
- Fix: WaterButler now handles Range header requests correctly!  There was an off-by-
↪one error where
WB was returning one byte more than requested.  This manifested in Safari being␣
↪unable to load
videos rendered by MFR.  Safari requests the first two bytes of a video before␣
↪issuing a full
request.  When WB returns three bytes, Safari refuses to try to render the file. ␣
↪Safari can now
render videos from most providers that support Range headers.  Unfortunately, it␣
↪still struggles
```

(continues on next page)

(continued from previous page)

```
with osfstorage for reasons that are still being investigated.  (thanks, @abought and␣
↪@johetordoff!)
- Feature: WaterButler now logs download and download-as-zip requests to the callback␣
↪given by the
auth provider.  This is intended to allow the OSF (the default auth provider) to␣
↪better log download
counts for files.  If WB detects that a request originates from MFR, it will relay␣
↪that on to the
callback, so the auth provider can account for that when compiling statistics.
- Feature: Add a new limited provider for Google Cloud Storage!  A limited provider␣
↪is one that can
be used as a data-storage backend provider for osfstorage.  It does NOT have a␣
↪corresponding OSF
addon interface, nor does it support the full range of WB actions.  It has been␣
↪tested locally, but
has yet to be tested in a staging or production environment, so EARLY ADOPT AT YOUR␣
↪OWN RISK!
- Fix: Stop installing aiohttpretty in editable mode to avoid containers hanging␣
↪while awaiting user
input.

0.37.1 (2018-02-06)
===================
- Feature: Increase default move/copy task timeout to 20s.  WaterButler will now wait␣
↪20 seconds for
move/copy tasks to complete before backgrounding the task and returning a 202␣
↪Accepted.

0.37.0 (2018-01-26)
===================
- ANNOUNCEMENT!  The WaterButler v0 API is once again DEPRECATED!  COS has removed␣
↪the last remnants
of it from the OSF (thanks, @alexschiller!) and has moved entirely to the v1 API.  It␣
↪will be
removed in the first release after *April 1*.  If you depend on v0, please get in␣
↪contact with us
(contact@cos.io) before then.
- Feature: Don't re-compress already-zipped files when downloading a directory as a␣
↪zip.  Re-zipping
wastes CPU time and will tend to result in larger zips overall.  (thanks,␣
↪@johnetordoff!)
- Feature: Add a unique request ID to WaterButler's response headers to help with␣
↪tracking down
errors.  (thanks, @johnetordoff!)
- Fix: Use the correct revision parameter name for fetching Amazon S3 revisions.  S3␣
↪revisions are
an optional feature that must be turned on via the AWS interface.  (thanks,␣
↪@TomBaxter!)
- Fix: Allow the GitHub to delete a folder in the repository root when it's the only␣
↪remaining
object.  (thanks, @TomBaxter!)
- Fix: Purge osfstorage uploads from the pending directory when the upload to the␣
↪backend storage
provider fails.  (thanks, @johnetordoff!)
- Fix: Return a 400 Bad Request when a user tries to copy a root folder to another␣
↪provider without
setting the `rename` parameter.  (thanks, @AddisonSchiller!)
```

(continues on next page)

(continued from previous page)

```
- Fix: Don't send invalid payloads to WaterButler's metrics tracking service. ␣
↪(thanks,
@AddisonSchiller!)
- Fix: Compute correct name for nested folders in the filesystem provider.  (thanks,
@AddisonSchiller & @johnetordoff!)
- Fix: Remove obsolete and unused `unsizable` flag from ResponseStreamReader. ␣
↪(thanks,
@AddisonSchiller & @johnetordoff!)
- Code: Upgrade the Dropbox provider to only use Dropbox's v2 endpoints.  (thanks,␣
↪@johnetordoff!)
- Code: Update WaterButler to support setuptools versions greater than v30.4.0.␣
↪WaterButler's
`__version__` declaration has moved from `waterbutler.__init__` to `waterbutler.
↪version`.  (thanks,
@johnetordoff!)
- Code: Distinguish between provider actions and authentication actions in the v1␣
↪move/copy code.

0.36.2 (2017-12-20)
===================
- Feature: Log osfstorage parity file metadata back to the OSF after upload.

0.36.1 (2017-12-11)
===================
- Fix: Update OneDrive metadata to report the correct materialized name.

0.36.0 (2017-12-05)
===================
- Feature: WaterButler now supports two new read-only providers, GitLab and Microsoft␣
↪OneDrive!
Read-only providers support browsing, downloading, downloading-as-zip, getting file␣
↪revision
history, and copying from connected repositories.  Thanks to the following devs for␣
↪their hard
work!
 - GitLab: @danielneis, @luismulinari, @rafaeldelucena
 - OneDrive: @caseyrygt, @alexandr-melnikov-dev-pro, @johnetordoff

0.35.0 (2017-11-13)
===================
- Feature: Allow copying from public resources with the OSF provider.  WaterButler␣
↪had been
requiring write permissions on the source resource for both moves and copies, but␣
↪copy only needs
read.  Update the v1 API to distinguish between the two types of requests.
- Docs: Document supported query parameters in the v1 API.
- Code: Improve test coverage for osfstorage and figshare.
- Code: Cleanups for Box, Google Drive, and GitHub providers.
- Code: Don't include test directories in module search paths.
- Code: Don't let query parameters override the HTTP verb in v1.

0.34.1 (2017-10-18)
===================
- Fix: Don't crash when a file on Google Drive is missing an md5 in its metadata. ␣
↪This occurs
for non-exportable files like Google Maps, Google Forms, etc.
```

(continues on next page)

(continued from previous page)

```
0.34.0 (2017-09-29)
===================
- ANNOUNCEMENT!  Sadly, the WaterButler v0 API is now *undeprecated*.  We've␣
↪discovered that the
OSF is still using it in a few places, so it's been given a temporary reprieve.  Once␣
↪those are
converted to v1, v0 will be re-deprecated then removed after an appropriate warning␣
↪period.
- Feature: For providers that return hashes on upload, WaterButler will calculate the␣
↪same hash
as the file streams and throw an informative error if its hash and the provider's␣
↪hash differ.
- Fix: Stop throwing exceptions when building exceptions to throw.  Pickled␣
↪exceptions are
resurrected in a peculiar fashion that some of WaterButler's exception classes could␣
↪not survive.
- Fix: Validate that the move/copy destination path is really a folder.
- Fix: Update the Box and Google Drive intra-{move,copy} actions to include children␣
↪in the
returned metadata for folders (and document it).
- Fix: Release Box responses on error.
- Code: Update the Postman test suite to include CRUD and move tests.
- Code: Start testing with python-3.6 on Travis.
- Code: Improve test coverage for all providers except osfstorage and figshare␣
↪(coming soon!).
- Code: Teach WaterButler to listen for a SIGTERM signal and exit immediately upon␣
↪receiving it.
This bypasses the 10 second wait for shutdown when running it in Docker.
- Code: Fix sphinx syntax errors in the WaterButler docs.

0.33.1 (2017-09-05)
===================
- Fix: Reject requests for Box IDs if the ID is valid, but the file or folder is␣
↪outside of the
project root.  (thanks, @AddisonSchiller!)

0.33.0 (2017-08-09)
===================
- ANNOUNCEMENT!  The WaterButler v0 API is DEPRECATED!  COS no longer uses it and has␣
↪moved
entirely to the v1 API.  It will be removed in the first release after October 1.  If␣
↪you depend on
v0, please get in contact with us (contact@cos.io) before then, and let us know.
- Feature: WaterButler now supports Bitbucket as a read-only provider!  As a read-
↪only provider,
you can browse, download, download-as-zip, get file revision history, and copy out of␣
↪your
connected Bitbucket repository.
- Feature: Sentry errors now include provider and resource as searchable parameters.␣
↪(thanks,
@abought!)
- Fix: Correctly describe the response of folder intra-move actions in GoogleDrive as␣
↪folders,
rather than files.
- Fix: WaterButler now correctly throttles multiple parallel requests.  The maximum␣
↪number of
simultaneous requests is set by the waterbutler.settings.OP_CONCURRENCY config␣
↪variable.
```

(continues on next page)

(continued from previous page)

```
0.32.3 (2017-07-20)
===================
- Fix: Quiet some overly-verbose error logging.

0.32.2 (2017-07-09)
===================
- Fix: Fix hanging figshare uploads by replacing a StreamReader.readexactly call with␣
↪a
StreamReader.read.  The underlying cause of this problem is still unknown.

0.32.1 (2017-07-07)
===================
- Fix: Correctly format the Last-Modified header returned from v1 HEAD requests. WB␣
↪had been
setting it to the datetime format used by the provider, but we should be following␣
↪the format laid
out by RFC 7232, S2.2.  (thanks, @icereval and @luizirber!)

0.32.0 (2017-06-14)
===================
- Fix: Send back the correct modified date when uploading a file to osfstorage.␣
↪osfstorage had
been sending back the modified date of the stored blob rather than the metadata from␣
↪the OSF.
- Fix: Support metadata and revisions for files shared on Google Drive with view- or␣
↪comment-only
permissions. Google Drive forbids access to the version-listing endpoint for these␣
↪sorts of files,
and WaterButler was not coping with that gracefully.
- Fix: Update WaterButler tests to work on Python >= v3.5.3. A change to coroutine␣
↪function
detection in Python v3.5.3 and v3.6.0 was causing tests to fail, as the mocked␣
↪coroutines were not
being properly unwrapped.
- Code: Add type annotations and a mypy test to the core WaterButler provider and the␣
↪box, dropbox,
googledrive, and figshare providers!  And lo, a new era of type-safety, strictness,␣
↪and peace was
ushered in, and its name was `inv mypy`. (thanks, @abought!)
- Code: Add support for code-coverage checking via coveralls.io. (thanks, @abought!)

0.31.1 (2017-06-01)
===================
- Fix: Fix OwnCloud issue that could result in folder creation outside the base␣
↪folder.  OwnCloud
was making assumptions about the formatting of the base folder that were not␣
↪necessarily true.

0.31.0 (2017-04-07)
===================
- Feature: Stop creating empty commits on GitHub when the requested action doesn't␣
↪change the tree
e.g. when updating a file with the exact same content as before.
- Fix: Moving or copying a folder within osfstorage will now return the metadata for␣
↪the folder's
children in the response.
```

(continues on next page)

(continued from previous page)

```
- Fix: Reject PATCH requests gracefully, instead of 500-ing.
- Fix: Disallow accessing Google Docs (.gdoc, .gsheet, etc.) without the extension.
- Fix: Fix poor error handling in Dropbox provider.
- Fix: Log WaitTimeoutErrors as log level info to Sentry. These are expected and␣
↪shouldn't be
considered full errors.

0.30.0 (2017-02-02)
===================
- Feature: Support the new London and Central Canada regions in Amazon S3. (thanks,␣
↪@johnetordoff!)
- Feature: Include provider-specific metrics in metric logging payloads, including␣
↪number of
requests issued.
- Fix: Don't crash when fetching file revision metadata from Google Drive.
- Fix: WaterButler docs are once again building on readthedocs.org! (thanks,␣
↪@johnetordoff!)
- Code: Update WaterButler to use invoke 0.13.0. If you have an existing checkout,␣
↪you will need to
upgrade invoke manually: pip install invoke==0.13.0 (thanks, @johnetordoff!)
- Code: Postman collections to test file and folder copy behavior for providers have␣
↪been added to
tests/postman/.  See docs/testing.rst for instructions on setting up and running them.
- Docs: WaterButler has been verified to work with python 3.5.3 and 3.6.0. From now␣
↪on, the docs
will mention which python versions WB has been verfied to work on. (thanks,␣
↪@johnetordoff!)

0.29.1 (2017-01-04)
===================
- Happy New Year!
- Fix: Be more ruthless about fixing setuptools breakage in Dockerfile. (thanks,␣
↪@cwisecarver!)

0.29.0 (2016-12-14)
===================
- Feature: WaterButler now uses the V2 APIs for both Dropbox and Figshare.
- Feature: Add a created timestamp to osfstorage file metadata.
- Feature: Support the new Mumbai and Ohio regions in Amazon S3. (thanks, @erinspace!)
- Feature: The server logs a message on startup, instead of just staring blankly at␣
↪you.
- Fix: Start appending extensions to Google Doc files to disambiguate identically-
↪named
files. e.g. foo.docx vs. foo.gsheet

0.28.1 (2016-12-13)
===================
- Pin setuptools to v30.4.0 to avoid package-namespace-related breakage.

0.28.0 (2016-10-31)
===================
- HALLOWEEN RELEASE! (^ ,-, ^)
- Feature: Download-as-zip now includes empty directories! (thanks, @darioncassel!)
- Feature: WaterButler now lists the full contents of Google Drive directories with␣
↪more than 1,000
children. (thanks, @TomBaxter!)
- Feature: WaterButler now lists the full contents of Box.com directories with more␣
↪than 1,000
```

(continues on next page)

(continued from previous page)

```
children. (thanks, @TomBaxter!)
- Fix: Teach ownCloud to be more efficient about moving and copying files with a␣
↪single provider.

0.27.1 (2016-10-24)
===================
- Fix: Fix broken Download-as-zip for GitHub by propagating the target branch during␣
↪recursive
traversal.
- Fix: Fix incorrectly detected self-overwrite when copying a file between the root␣
↪paths of two
separate Box.com accounts.

0.27.0 (2016-10-19)
===================
- Feature: Attempting to move or copy a file over itself will now fail with a 409␣
↪Conflict, even
across different resources.
- Fix: Fix bugs in ownCloud provider that were breaking renames.
- Fix: v1 metadata requests now accept the `version` and `revision` query parameters␣
↪like v0.

0.26.1 (2016-10-18)
===================
- Feature: Dockerized WaterButler can now take a commit sha from the environment to␣
↪indicate
version to deploy. (thanks, @icereval!)

0.26.0 (2016-10-11)
===================
- Feature: WaterButler now supports ownCloud as a full provider! (thanks, @kwierman!)
- Feature: WaterButler accepts configuration from the environment, overriding any␣
↪file-based
configuration.  This helps WB integrate nicer in a docker-compose environment.␣
↪(thanks, @icereval!)
- Fix: Gracefully handle branch-related GitHub errors.
- Code: Start labeling user-caused errors as level=info in Sentry (thanks, @TomBaxter!
↪)
- Code: Log redirect-based downloads to analytics.
- Code: Bump dependencies on Raven and cryptography.  cryptography v1.5.2 now␣
↪installs on OSX via
wheel.  This should silence scary-sounding cffi warnings!

0.25.0 (2016-09-22)
===================
- Feature: Include user id when requesting files from osfstorage, to allow the OSF to␣
↪distinguish
contributing users in download counts. (thanks, @darioncassel!)

0.24.0 (2016-09-14)
===================
- Feature: Update the v1 API to passthrough unrecognized query params to the provider.
- Feature: Teach the GitHub provider to accept branch identifiers in the URL and body␣
↪of v1
operations.  You can now do cross-branch move/copies with the v1 API!
- Feature: The GitHub provider now includes the branch operated on in its callback␣
↪logs.
```

(continues on next page)

(continued from previous page)

```
- Fix: Add `--pty` arguments to invoke install and invoke wheelhouse, to support␣
↪building WB Docker
images without pseudoterminals. (thanks, @emetsger!)

0.23.3 (2016-08-31)
===================
- Fix: Fix flake error in remote_logging. Not at all embarrassing.

0.23.2 (2016-08-31)
===================
- Fix: For analytics, convert byte sizes into more convenient units.
- Code: in sizes.py, call kilobytes "KBs", not "Bs"

0.23.1 (2016-08-31)
===================
- Fix: Dataverse changed their API file metadata repsonse format. Update provider to␣
↪handle both
formats.

0.23.0 (2016-08-25)
===================
- Code: Rewrite public file action logging to sync with MFR.
- Docs: Document intra_move and intra_copy in core provider.

0.22.1 (2016-08-19)
===================
- Fix: Don't try to derive modified_utc for osfstorage files that lack a modified␣
↪date.

0.22.0 (2016-08-19)
===================
- Feature: File metadata now includes a modified_utc field that is the modified␣
↪timestamp in
standard ISO-8601 format (YYYY-MM-DDTHH:MM::SS+00:00). (thanks, @TomBaxter!)
- Feature: Metadata for osfstorage files will contain the file GUID, once the OSF is␣
↪updated to
return that information. (thanks, @Johnetordoff!)
- Feature: WaterButler can now log file actions to Keen.io.
- Fix: core.utils.async_retry was always intended to be fire-and-forget, but wasn't␣
↪when await()-ed.
It will now run until done, instead of executing one retry per await.  The tests have␣
↪been updated
to match this behavior.
- Fix: Update copy and move celery tasks so that a failed logging callback will not␣
↪make them return
a 500.  Now they always return the result and metadata of the move/copy action.
- Fix: Logging callbacks are now retried five times, no matter where they are called␣
↪from.
- Fix: Update Dockerfile to register plugins.
- Fix: Correct minor typos and links to Python and aiohttp in the docs.

0.21.4 (2016-08-02)
===================
- Fix: Ask Box not to zip our download requests. (thanks, @caseyrygt!)

0.21.3 (2016-08-01)
===================
```

(continues on next page)

(continued from previous page)

```
- Fix: Bump wheel dep to 0.26.0 to fix travis build.

0.21.2 (2016-08-01)
===================
- Fix: Pin cryptography to v1.3.4 to avoid v1,4 incompatibilities with OS X's vendor␣
↪openssl.
(thanks, @erinspace!)

0.21.1 (2016-07-12)
===================
- Fix: Stop duplicating parent folder when searching Google Drive for Google docs. ␣
↪(thanks,
@TomBaxter!)

0.21.0 (2016-06-16)
===================
- Feature: Allow cross origin requests when an Authorization header is provided␣
↪without
cookies.  (thanks, @samchrisinger!)
- Feature: Don't set any CORS headers if Origin is not provided (e.g. non-browser␣
↪client)
- Fix: v0's copy action now checks can_intra_copy instead of can_intra_move.
- Fix: Stop sending requests to GitHub with the `application/vnd.github.VERSION.raw`␣
↪media-type.
Start sending them `application/vnd.github.v3.raw`.
- Fix: Our API docs had typos in the example URLs.  For shame.
- Code: Refactor logging callback into one location.  Previously, v0 creates, v0 move/
↪copies, v1
actions, and the move/copy celery tasks each had their own bespoke code for logging␣
↪actions to the
authorizer-provided callback.  All of that has been merged into one method with a␣
↪common interface.
This shouldn't have any user-visible changes, but it will make the developers' lives␣
↪much easier.
- Code: Remove unused code from googledrive provider
- Code: WaterButlerPath learned `materialized_path()` (a.k.a its __str__␣
↪representation).

0.20.4 (2016-06-16)
===================
- Finish support for zipfiles > 4Gb.  Large zipfiles should now be uncompressible by
double-clicking on the zipfiles in OS X, Windows, and Linux.  On OS X, /usr/bin/ditto␣
↪and
The Unarchiver have been confirmed to work. /usr/bin/unzip will *not* work, as the␣
↪version
they include does not have Zip64 support.
- Update the install docs to pin invoke to 0.11.1.

0.20.3 (2016-06-13)
===================
- Release fixes for deployment and intermediate DAZ > 4Gb fixes.

0.20.2 (2016-06-13)
===================
- Pin invoke to v0.11.1.  Our tasks.py is incompatible with v0.13.

0.20.1 (2016-06-13)
```

(continues on next page)

(continued from previous page)

```
==================
- Try fixing Download-As-Zip for resulting zipfiles > 4Gb.  The default zip format␣
↪only supports
up to 4Gb files, but the zip64 extension can handle much larger sizes.  WB hand-rolls␣
↪its zips,
so the zip64 format has to be constructed manually.  Unfortunately, the fixes applied␣
↪in 0.20.1
were not enough.  The remaining updates were added in 0.20.4.
- Add a Dockerfile to simplify running WaterButler in dev environments.
- Pin some dependencies and update our travis config to avoid spurious build failures.

0.20.0 (2016-04-29)
==================
- Fix Download-As-Zip for Google Drive to set the correct extensions for exported␣
↪Google Doc
files.
- Add 'resource' to V1 response.
- Minor doc updates: Add urls to external API documentation and note provider quirks.

0.19.5 (2016-04-18)
==================
- Brown Paper Bag release: **REALLY** fix syntax error in Github's Unsupported Repo␣
↪exception.
The 0.19.4 release fixed nothing.  I (@felliott) am an idiot.

0.19.4 (2016-04-18)
==================
- Fix syntax error in Github's Unsupported Repo exception.

0.19.3 (2016-04-15)
==================
- Increase number of files returned from a Box directory from 50 to 1000. (thanks,␣
↪@TomBaxter!)
- Exclude Google Maps from Google Drive listing.  Google doesn't provide a way to␣
↪export maps,
so exclude them from listing / downloading for now.  This is the same behavior as␣
↪Google Forms.

0.19.2 (2016-04-14)
==================
- Make v1 move/copy return file metadata in its logging payload, so the OSF can track␣
↪files
across providers.

0.19.1 (2016-04-13)
==================
- Update boto dependency to get fixes for large file uploads to Amazon Glacier.␣
↪(thanks,
@caileyfitz, for your patient testing!)
- Increase number of files returned from a GDrive directory listing from 100 to 1000.

0.19.0 (2016-04-04)
==================
- Feature: WaterButler now runs on Python 3.5!  A major speed boost should be
noticeable, especially for zipped folder downloads.  All hail @chrisseto and
@TomBaxter for seeing this through!
- Feature: Zipped folder downloads now use the folder name as the zip filename.  If␣
↪the folder
```

(continues on next page)

(continued from previous page)

```
is the storage root, the name is `$provider.zip` (e.g. `googledrive.zip`).
- Code: WaterButlerPath has a new `is_folder` method. It's the same as `is_dir`.
- Code: waterbutler.core.BaseProvider learned `path_from_metadata`.

0.18.4 (2016-04-01)
===================
- Fix: Bump PyJWE dependency to 1.0.0 to match with OSF v0.66.0

0.18.3 (2016-03-28)
===================
- Fix: Renaming a Google doc (.gdoc, .gsheet) file should not and no longer truncates␣
↪the new
filename to just the first letter.

0.18.2 (2016-03-17)
===================
- The "Leprechauns Ate My Blob" emergency release!
- Fix: Don't throw an error if the github file being requested is larger than 1Mb. ␣
↪There will
be a more correct fix in the next minor release.

0.18.1 (2016-03-15)
===================
- Fix: Stop crashing if the `Content-Length` header is not set on a v1 folder create␣
↪request.
A missing `Content-Length` is fine for folder create requests.

0.18.0 (2016-03-09)
===================
- BREAKING v1 API CHANGE (sortof):  Updating a file by issuing a PUT to its parent␣
↪folder and
passing its name as a query parameter is no longer supported and will now throw a 409␣
↪Conflict.
This is still the correct way to create a file, but updating must be done by issuing␣
↪the PUT to
the file's own endpoint.  This was supposed to be fixed back in December, but I␣
↪(@felliott) did
a **very** poor job of it, meaning some providers still allowed it.  The API␣
↪documentation has
been updated to match.  If you use the `/links/upload` action from the JSON-API␣
↪response, you
**DO NOT** need to update your code,  That link is already correct.
- Feature: DELETEing a file or folder in the GDrive provider now sends it to the␣
↪trash instead
or hard-deleting it.
- Feature: Issuing a DELETE to the storage root of a provider will now clear out its␣
↪content,
but not delete the storage root itself.  This was undefined behavior before.  Some␣
↪providers
would disallow it, some would crash, others would do the right thing.  This is now␣
↪officially
supported.  To make sure the file contents are not wiped out on accident, the query␣
↪parameter
`confirm_delete=1` must be passed when clearing the contents.  Otherwise, WB will␣
↪return a 400.
- Fix: GoogleDrive provider now returns 201 Created when creating and 200 OK when␣
↪overwriting
```

(continues on next page)

(continued from previous page)

```
during copy operations.
- Fix: Copying / moving empty folders into a directory with a similarly named folder␣
↪with 'keep
both' semantics no longer overwrites old folder and properly increments the new file␣
↪name.
- Fix: Always set `kind` parameter for files and folders in v1 logging callback to␣
↪avoid crashing
the OSF waterbutler logging endpoint.
- Fix: For 'keep both' conflict semantics when more than one incremented version␣
↪already exists.
- Fix: Github move/copy folder requests with 'replace' semantics now correctly␣
↪overwrites the
target folder, rather than merging the contents.
- Docs: @TomBaxter++ has added more docstrings to the base provider and has started␣
↪documenting
the quirks of our existing providers.
- Docs: v0.18.0 WaterButler DO NOT work with python 3.5.  3.4 is required.  Mention␣
↪this.
- Docs: The Center for Open Science is hiring! (thanks, @andrewsallans!)

0.17.0 (2016-02-29)
===================
- LEAP DAY RELEASE!
- Feature: Add throttling to make_request!   Some hosts don't like it when we fire␣
↪off too many
requests at once.  Now we limit it to 10 requests / second and a maximum of 5␣
↪concurrently.  To
adjust the rates, update the REQUEST_LIMIT and OP_CONCURRENCY attributes in settings.
↪py. (thanks
@chrisseto!)
- Fix: Update dropbox API urls.

0.16.5 (2016-02-22)
===================
- No changes. Applied and reverted throttling patches.

0.16.4 (2016-02-18)
===================
- Fix: Add semaphore to make_request to limit concurrent outgoing requests to 25.

0.16.3 (2016-02-14)
===================
- Fix: Apply certificate fix in proper scope.

0.16.2 (2016-02-14)
===================
- Fix: Patch to prevent certification verification failure w/ rackspace services.

0.16.1 (2016-02-11)
===================
- Fix: Properly freeze time in tests to avoid spurious test failures.
- Update Sentry Raven client now that it has core asyncio support.

0.16.0 (2016-02-03)
===================
- Feature: Support S3 buckets with periods in their name for non-US East
regions.
```

(continues on next page)

(continued from previous page)

```
- Feature: Started filling out and reorganizng the dev docs!  The v1 API is
now documented, and waterbutler.core.metadata, waterbutler.core.path have
a lot more docstrings.  A skeleton overview of WaterButler is available.  More
to come...
- Fix: Make the filesystem provider declare timezones for their modified date.
- Fix: Update FigShare's web view URL for viewing unpublished files.

0.15.1 (2016-01-21)
===================
- Fix incorrect logging of paths for move/copy task and v0 deletes
- Fix incorrect path calculation for cross-resource move/copys via Dropbox
- Properly encode googledrive path in log payloads

0.15.0 (2016-01-15)
===================
- Enforce V1 path semantics. Folders must have trailing slash, files must NOT
have trailing slash.  Failing to abide results in a 404
- Allow creating a file or folder in a directory with an identically-named
entity of the opposite type, but only for those providers that allow it.
- Fix recursive delete via s3 when folder name has special characters
- Fix multiple 500s on github provider
- Fix many v1 logging issues
- Clarify install instructions, (thanks @rafaeldelucena!)
- Add `clean` task to remove old .pyc files

0.14.0 (2015-11-05)
===================
- Update to a python3.5 compliant version of invoke
- Raise proper exceptions for github repos with too many fields
- Updates for OSFStorage file checkout
- Clean up JSON API Responses

0.13.0 (2015-10-08)
===================
- waterbutler API v1 now returns JSON API formatted data
- DEBUG is now an option for waterbutler root settings
- OSF auth handler now authenticates via JWTs
- Moves and copies done via v1 will now return a 409 rather than implicitly␣
↪overwriting
- Failed log callbacks are now logged
- Various smaller fixes

0.12.0 (2015-09-17)
===================
- waterbutler.server
  - Restructured into API version modules
  - API v1 has been implemented
    - Only one endpoint exists /v1/resources/<>/providers/<>/<path>
  - API v0 is now deprecated
  - Callbacks will be retried if they do not get a 200 response

- waterbutler.core
  - Invalid providers are now handled properly
  - WaterbutlerPath now has an identifier_path property for id based backends
  - Revision is now an accepted parameter of Provider#metadata and Provider#download

- Github now returns modifed dates when available
```

(continues on next page)

(continued from previous page)

```
- Google drive's title queries now only use single quotes (')
- OsfStorage's validate_path function now works properly
- Osfstorage now properly responds created to internal copies

0.11.0 (2015-08-31)
===================
- OsfStorage now returns hashes

0.10.0 (2015-08-10)
===================
- Allow S3 uploads to be encrypted at rest via S3's API

0.9.0 (2015-07-29)
==================

- Web view links are included in the extra field when available
- Add many a test for moving and copying, tasks and endpoints
- Allow OsfStorage tasks to be disabled by adding including archive: false

0.8.0 (2015-07-14)
==================

- Add support for passing the Range head through
- Exceptions are no longer raised when a client connection cuts off early
- ResponseStreamReader may override file names via .name
- Calls to metadata now returns BaseMetadata objects or a list thereof
- Upgrade to tornado 4.2, which increases compatability with asyncio
- General code clean up
- Add a style/contributing guide
- Uploading files is now implemented with unix sockets and will not buffer the
entire file into memory
- Accept files up to 4.9GBs
- view_url is included with file metadata requests
- Flake8 is now much more aggressive
- General code clean up

0.7.0 (2015-06-18)
==================

- Read me updates
- Various fixes for S3
- Fixes to Dataverse's copy and move
- Various fixes for figshare

0.6.0 (2015-06-07)
==================

- Various fixes to Google drive
- Allow response streams to be "unsizable"
- Return an additional "etag" field with file metadata

0.5.0 (2015-05-25)
==================

- Implement moving and expose it via http
- Implement copying and expose it via http
- Implement downloading as zip and expose it via http
```

(continues on next page)

(continued from previous page)

```
0.4.0 (2015-04-28)
==================

- Add folder creation

0.3.0 (2015-04-20)
==================

- Add harvard dataverse as a provider

0.2.4 (2015-03-18)
==================

- Allow ssl certs to be specified in the config
```

3.3 License

```
                                 Apache License
                           Version 2.0, January 2004
                        http://www.apache.org/licenses/

TERMS AND CONDITIONS FOR USE, REPRODUCTION, AND DISTRIBUTION

1. Definitions.

   "License" shall mean the terms and conditions for use, reproduction,
   and distribution as defined by Sections 1 through 9 of this document.

   "Licensor" shall mean the copyright owner or entity authorized by
   the copyright owner that is granting the License.

   "Legal Entity" shall mean the union of the acting entity and all
   other entities that control, are controlled by, or are under common
   control with that entity. For the purposes of this definition,
   "control" means (i) the power, direct or indirect, to cause the
   direction or management of such entity, whether by contract or
   otherwise, or (ii) ownership of fifty percent (50%) or more of the
   outstanding shares, or (iii) beneficial ownership of such entity.

   "You" (or "Your") shall mean an individual or Legal Entity
   exercising permissions granted by this License.

   "Source" form shall mean the preferred form for making modifications,
   including but not limited to software source code, documentation
   source, and configuration files.

   "Object" form shall mean any form resulting from mechanical
   transformation or translation of a Source form, including but
   not limited to compiled object code, generated documentation,
   and conversions to other media types.

   "Work" shall mean the work of authorship, whether in Source or
```

(continues on next page)

(continued from previous page)

```
   Object form, made available under the License, as indicated by a
   copyright notice that is included in or attached to the work
   (an example is provided in the Appendix below).

   "Derivative Works" shall mean any work, whether in Source or Object
   form, that is based on (or derived from) the Work and for which the
   editorial revisions, annotations, elaborations, or other modifications
   represent, as a whole, an original work of authorship. For the purposes
   of this License, Derivative Works shall not include works that remain
   separable from, or merely link (or bind by name) to the interfaces of,
   the Work and Derivative Works thereof.

   "Contribution" shall mean any work of authorship, including
   the original version of the Work and any modifications or additions
   to that Work or Derivative Works thereof, that is intentionally
   submitted to Licensor for inclusion in the Work by the copyright owner
   or by an individual or Legal Entity authorized to submit on behalf of
   the copyright owner. For the purposes of this definition, "submitted"
   means any form of electronic, verbal, or written communication sent
   to the Licensor or its representatives, including but not limited to
   communication on electronic mailing lists, source code control systems,
   and issue tracking systems that are managed by, or on behalf of, the
   Licensor for the purpose of discussing and improving the Work, but
   excluding communication that is conspicuously marked or otherwise
   designated in writing by the copyright owner as "Not a Contribution."

   "Contributor" shall mean Licensor and any individual or Legal Entity
   on behalf of whom a Contribution has been received by Licensor and
   subsequently incorporated within the Work.

2. Grant of Copyright License. Subject to the terms and conditions of
   this License, each Contributor hereby grants to You a perpetual,
   worldwide, non-exclusive, no-charge, royalty-free, irrevocable
   copyright license to reproduce, prepare Derivative Works of,
   publicly display, publicly perform, sublicense, and distribute the
   Work and such Derivative Works in Source or Object form.

3. Grant of Patent License. Subject to the terms and conditions of
   this License, each Contributor hereby grants to You a perpetual,
   worldwide, non-exclusive, no-charge, royalty-free, irrevocable
   (except as stated in this section) patent license to make, have made,
   use, offer to sell, sell, import, and otherwise transfer the Work,
   where such license applies only to those patent claims licensable
   by such Contributor that are necessarily infringed by their
   Contribution(s) alone or by combination of their Contribution(s)
   with the Work to which such Contribution(s) was submitted. If You
   institute patent litigation against any entity (including a
   cross-claim or counterclaim in a lawsuit) alleging that the Work
   or a Contribution incorporated within the Work constitutes direct
   or contributory patent infringement, then any patent licenses
   granted to You under this License for that Work shall terminate
   as of the date such litigation is filed.

4. Redistribution. You may reproduce and distribute copies of the
   Work or Derivative Works thereof in any medium, with or without
   modifications, and in Source or Object form, provided that You
   meet the following conditions:
```

(continues on next page)

(continued from previous page)

```
   (a) You must give any other recipients of the Work or
       Derivative Works a copy of this License; and

   (b) You must cause any modified files to carry prominent notices
       stating that You changed the files; and

   (c) You must retain, in the Source form of any Derivative Works
       that You distribute, all copyright, patent, trademark, and
       attribution notices from the Source form of the Work,
       excluding those notices that do not pertain to any part of
       the Derivative Works; and

   (d) If the Work includes a "NOTICE" text file as part of its
       distribution, then any Derivative Works that You distribute must
       include a readable copy of the attribution notices contained
       within such NOTICE file, excluding those notices that do not
       pertain to any part of the Derivative Works, in at least one
       of the following places: within a NOTICE text file distributed
       as part of the Derivative Works; within the Source form or
       documentation, if provided along with the Derivative Works; or,
       within a display generated by the Derivative Works, if and
       wherever such third-party notices normally appear. The contents
       of the NOTICE file are for informational purposes only and
       do not modify the License. You may add Your own attribution
       notices within Derivative Works that You distribute, alongside
       or as an addendum to the NOTICE text from the Work, provided
       that such additional attribution notices cannot be construed
       as modifying the License.

   You may add Your own copyright statement to Your modifications and
   may provide additional or different license terms and conditions
   for use, reproduction, or distribution of Your modifications, or
   for any such Derivative Works as a whole, provided Your use,
   reproduction, and distribution of the Work otherwise complies with
   the conditions stated in this License.

5. Submission of Contributions. Unless You explicitly state otherwise,
   any Contribution intentionally submitted for inclusion in the Work
   by You to the Licensor shall be under the terms and conditions of
   this License, without any additional terms or conditions.
   Notwithstanding the above, nothing herein shall supersede or modify
   the terms of any separate license agreement you may have executed
   with Licensor regarding such Contributions.

6. Trademarks. This License does not grant permission to use the trade
   names, trademarks, service marks, or product names of the Licensor,
   except as required for reasonable and customary use in describing the
   origin of the Work and reproducing the content of the NOTICE file.

7. Disclaimer of Warranty. Unless required by applicable law or
   agreed to in writing, Licensor provides the Work (and each
   Contributor provides its Contributions) on an "AS IS" BASIS,
   WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or
   implied, including, without limitation, any warranties or conditions
   of TITLE, NON-INFRINGEMENT, MERCHANTABILITY, or FITNESS FOR A
   PARTICULAR PURPOSE. You are solely responsible for determining the
```

(continues on next page)

(continued from previous page)

```
   appropriateness of using or redistributing the Work and assume any
   risks associated with Your exercise of permissions under this License.

8. Limitation of Liability. In no event and under no legal theory,
   whether in tort (including negligence), contract, or otherwise,
   unless required by applicable law (such as deliberate and grossly
   negligent acts) or agreed to in writing, shall any Contributor be
   liable to You for damages, including any direct, indirect, special,
   incidental, or consequential damages of any character arising as a
   result of this License or out of the use or inability to use the
   Work (including but not limited to damages for loss of goodwill,
   work stoppage, computer failure or malfunction, or any and all
   other commercial damages or losses), even if such Contributor
   has been advised of the possibility of such damages.

9. Accepting Warranty or Additional Liability. While redistributing
   the Work or Derivative Works thereof, You may choose to offer,
   and charge a fee for, acceptance of support, warranty, indemnity,
   or other liability obligations and/or rights consistent with this
   License. However, in accepting such obligations, You may act only
   on Your own behalf and on Your sole responsibility, not on behalf
   of any other Contributor, and only if You agree to indemnify,
   defend, and hold each Contributor harmless for any liability
   incurred by, or claims asserted against, such Contributor by reason
   of your accepting any such warranty or additional liability.

END OF TERMS AND CONDITIONS

APPENDIX: How to apply the Apache License to your work.

   To apply the Apache License to your work, attach the following
   boilerplate notice, with the fields enclosed by brackets "[]"
   replaced with your own identifying information. (Don't include
   the brackets!)  The text should be enclosed in the appropriate
   comment syntax for the file format. We also recommend that a
   file or class name and description of purpose be included on the
   same "printed page" as the copyright notice for easier
   identification within third-party archives.

Copyright 2013-2014 Center for Open Science

Licensed under the Apache License, Version 2.0 (the "License");
you may not use this file except in compliance with the License.
You may obtain a copy of the License at

    http://www.apache.org/licenses/LICENSE-2.0

Unless required by applicable law or agreed to in writing, software
distributed under the License is distributed on an "AS IS" BASIS,
WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.
See the License for the specific language governing permissions and
limitations under the License.
```

- genindex
- modindex
- search

Python Module Index

W

waterbutler, 19
waterbutler.constants, 17
waterbutler.core, 21
waterbutler.core.auth, 19
waterbutler.core.logging, 19
waterbutler.core.metrics, 19
waterbutler.core.signing, 21
waterbutler.providers, 31
waterbutler.server, 32
waterbutler.server.settings, 31
waterbutler.server.utils, 31
waterbutler.settings, 17
waterbutler.sizes, 18
waterbutler.version, 18

Index

A

add() (*waterbutler.core.metrics.MetricsBase method*), 20
add_field() (*waterbutler.core.streams.FormDataStream method*), 29
add_fields() (*waterbutler.core.streams.FormDataStream method*), 29
add_file() (*waterbutler.core.streams.FormDataStream method*), 29
add_reader() (*waterbutler.core.streams.BaseStream method*), 22
add_reader() (*waterbutler.core.streams.FileStreamReader method*), 25
add_reader() (*waterbutler.core.streams.RequestStreamReader method*), 23
add_reader() (*waterbutler.core.streams.ResponseStreamReader method*), 22
add_reader() (*waterbutler.core.streams.StringStream method*), 26
add_streams() (*waterbutler.core.streams.FormDataStream method*), 30
add_streams() (*waterbutler.core.streams.MultiStream method*), 28
add_writer() (*waterbutler.core.streams.BaseStream method*), 22
add_writer() (*waterbutler.core.streams.FileStreamReader method*), 25
add_writer() (*waterbutler.core.streams.RequestStreamReader method*), 23
add_writer() (*waterbutler.core.streams.ResponseStreamReader method*), 22
add_writer() (*waterbutler.core.streams.StringStream method*), 26
append() (*waterbutler.core.metrics.MetricsBase method*), 20
at_eof() (*waterbutler.core.streams.FileStreamReader method*), 25
at_eof() (*waterbutler.core.streams.FormDataStream method*), 30
at_eof() (*waterbutler.core.streams.MultiStream method*), 28
at_eof() (*waterbutler.core.streams.RequestStreamReader method*), 24
at_eof() (*waterbutler.core.streams.ResponseStreamReader method*), 22
at_eof() (*waterbutler.core.streams.StringStream method*), 26
AuthType (*class in waterbutler.core.auth*), 19

B

BaseAuthHandler (*class in waterbutler.core.auth*), 19
BaseStream (*class in waterbutler.core.streams*), 21
bytes_downloaded (*waterbutler.server.utils.UtilMixin attribute*), 32
bytes_uploaded (*waterbutler.server.utils.UtilMixin attribute*), 32

C

can_write_eof() (*waterbutler.core.streams.HashStreamWriter method*), 26
child() (*in module waterbutler.settings*), 18
child() (*waterbutler.settings.SettingsDict method*), 18
chunk_reader() (*waterbutler.core.streams.FileStreamReader method*), 25
close() (*waterbutler.core.streams.FileStreamReader method*), 25

close() (*waterbutler.core.streams.HashStreamWriter method*), 26
content_range (*waterbutler.core.streams.ResponseStreamReader attribute*), 22
content_type (*waterbutler.core.streams.ResponseStreamReader attribute*), 22
CORsMixin (*class in waterbutler.server.utils*), 32
CutoffStream (*class in waterbutler.core.streams*), 31

D

DESTINATION (*waterbutler.core.auth.AuthType attribute*), 19
digest (*waterbutler.core.streams.HashStreamWriter attribute*), 26

E

end_boundary (*waterbutler.core.streams.FormDataStream attribute*), 29
exception() (*waterbutler.core.streams.FileStreamReader method*), 25
exception() (*waterbutler.core.streams.FormDataStream method*), 30
exception() (*waterbutler.core.streams.MultiStream method*), 28
exception() (*waterbutler.core.streams.RequestStreamReader method*), 24
exception() (*waterbutler.core.streams.ResponseStreamReader method*), 22
exception() (*waterbutler.core.streams.StringStream method*), 26

F

feed_data() (*waterbutler.core.streams.FileStreamReader method*), 25
feed_data() (*waterbutler.core.streams.FormDataStream method*), 30
feed_data() (*waterbutler.core.streams.MultiStream method*), 28
feed_data() (*waterbutler.core.streams.RequestStreamReader method*), 24
feed_data() (*waterbutler.core.streams.ResponseStreamReader method*), 22
feed_data() (*waterbutler.core.streams.StringStream method*), 26
feed_eof() (*waterbutler.core.streams.BaseStream method*), 22
feed_eof() (*waterbutler.core.streams.FileStreamReader method*), 25
feed_eof() (*waterbutler.core.streams.FormDataStream method*), 30
feed_eof() (*waterbutler.core.streams.MultiStream method*), 28
feed_eof() (*waterbutler.core.streams.RequestStreamReader method*), 24
feed_eof() (*waterbutler.core.streams.ResponseStreamReader method*), 22
feed_eof() (*waterbutler.core.streams.StringStream method*), 27
fetch() (*waterbutler.core.auth.BaseAuthHandler method*), 19
FileStreamReader (*class in waterbutler.core.streams*), 25
finalize() (*waterbutler.core.streams.FormDataStream method*), 29
format() (*waterbutler.core.logging.MaskFormatter method*), 19
FormDataStream (*class in waterbutler.core.streams*), 29
full_key() (*waterbutler.settings.SettingsDict method*), 18

G

get() (*waterbutler.core.auth.BaseAuthHandler method*), 19
get() (*waterbutler.settings.SettingsDict method*), 18
get_bool() (*waterbutler.settings.SettingsDict method*), 18
get_nullable() (*waterbutler.settings.SettingsDict method*), 18
get_object() (*waterbutler.settings.SettingsDict method*), 18

H

HashStreamWriter (*class in waterbutler.core.streams*), 26
headers (*waterbutler.core.streams.FormDataStream attribute*), 29
hexdigest (*waterbutler.core.streams.HashStreamWriter attribute*), 26

I

incr() (*waterbutler.core.metrics.MetricsBase method*), 20

K

key (*waterbutler.core.metrics.MetricsRecord attribute*), 20
key (*waterbutler.core.metrics.MetricsSubRecord attribute*), 20
key() (*waterbutler.core.metrics.MetricsBase method*), 20

M

make_boundary() (*waterbutler.core.streams.FormDataStream class method*), 29
make_header() (*waterbutler.core.streams.FormDataStream class method*), 29
manifesto() (*waterbutler.core.metrics.MetricsBase method*), 20
MaskFormatter (*class in waterbutler.core.logging*), 19
merge() (*waterbutler.core.metrics.MetricsBase method*), 20
MetricsBase (*class in waterbutler.core.metrics*), 19
MetricsRecord (*class in waterbutler.core.metrics*), 20
MetricsSubRecord (*class in waterbutler.core.metrics*), 20
MultiStream (*class in waterbutler.core.streams*), 28

N

name (*waterbutler.core.streams.ResponseStreamReader attribute*), 22
new_subrecord() (*waterbutler.core.metrics.MetricsRecord method*), 20
new_subrecord() (*waterbutler.core.metrics.MetricsSubRecord method*), 21

O

options() (*waterbutler.server.utils.CORsMixin method*), 32
order_recursive() (*in module waterbutler.core.signing*), 21

P

parse_request_range() (*in module waterbutler.server.utils*), 31
partial (*waterbutler.core.streams.ResponseStreamReader attribute*), 22

R

read() (*waterbutler.core.streams.BaseStream method*), 22
read() (*waterbutler.core.streams.CutoffStream method*), 31
read() (*waterbutler.core.streams.FileStreamReader method*), 25
read() (*waterbutler.core.streams.FormDataStream method*), 30
read() (*waterbutler.core.streams.MultiStream method*), 28
read() (*waterbutler.core.streams.RequestStreamReader method*), 24
read() (*waterbutler.core.streams.ResponseStreamReader method*), 22
read() (*waterbutler.core.streams.StringStream method*), 27
readexactly() (*waterbutler.core.streams.FileStreamReader method*), 25
readexactly() (*waterbutler.core.streams.FormDataStream method*), 30
readexactly() (*waterbutler.core.streams.MultiStream method*), 28
readexactly() (*waterbutler.core.streams.RequestStreamReader method*), 24
readexactly() (*waterbutler.core.streams.ResponseStreamReader method*), 23
readexactly() (*waterbutler.core.streams.StringStream method*), 27
readline() (*waterbutler.core.streams.FileStreamReader method*), 25
readline() (*waterbutler.core.streams.FormDataStream method*), 30
readline() (*waterbutler.core.streams.MultiStream method*), 28
readline() (*waterbutler.core.streams.RequestStreamReader method*), 24
readline() (*waterbutler.core.streams.ResponseStreamReader method*), 23
readline() (*waterbutler.core.streams.StringStream method*), 27
readuntil() (*waterbutler.core.streams.FileStreamReader method*), 26
readuntil() (*waterbutler.core.streams.FormDataStream method*),

30
readuntil() (*waterbutler.core.streams.MultiStream method*), 28
readuntil() (*waterbutler.core.streams.RequestStreamReader method*), 24
readuntil() (*waterbutler.core.streams.ResponseStreamReader method*), 23
readuntil() (*waterbutler.core.streams.StringStream method*), 27
remove_reader() (*waterbutler.core.streams.BaseStream method*), 22
remove_reader() (*waterbutler.core.streams.FileStreamReader method*), 26
remove_reader() (*waterbutler.core.streams.RequestStreamReader method*), 24
remove_reader() (*waterbutler.core.streams.ResponseStreamReader method*), 23
remove_reader() (*waterbutler.core.streams.StringStream method*), 27
remove_writer() (*waterbutler.core.streams.BaseStream method*), 22
remove_writer() (*waterbutler.core.streams.FileStreamReader method*), 26
remove_writer() (*waterbutler.core.streams.RequestStreamReader method*), 25
remove_writer() (*waterbutler.core.streams.ResponseStreamReader method*), 23
remove_writer() (*waterbutler.core.streams.StringStream method*), 27
RequestStreamReader (*class in waterbutler.core.streams*), 23
ResponseStreamReader (*class in waterbutler.core.streams*), 22

S

serialize() (*waterbutler.core.metrics.MetricsBase method*), 20
serialize() (*waterbutler.core.metrics.MetricsRecord method*), 20
serialize_payload() (*in module waterbutler.core.signing*), 21
set_default_headers() (*waterbutler.server.utils.CORsMixin method*), 32
set_exception() (*waterbutler.core.streams.FileStreamReader method*), 26
set_exception() (*waterbutler.core.streams.FormDataStream method*), 30
set_exception() (*waterbutler.core.streams.MultiStream method*), 29
set_exception() (*waterbutler.core.streams.RequestStreamReader method*), 25
set_exception() (*waterbutler.core.streams.ResponseStreamReader method*), 23
set_exception() (*waterbutler.core.streams.StringStream method*), 27
set_status() (*waterbutler.server.utils.UtilMixin method*), 32
set_transport() (*waterbutler.core.streams.FileStreamReader method*), 26
set_transport() (*waterbutler.core.streams.FormDataStream method*), 31
set_transport() (*waterbutler.core.streams.MultiStream method*), 29
set_transport() (*waterbutler.core.streams.RequestStreamReader method*), 25
set_transport() (*waterbutler.core.streams.ResponseStreamReader method*), 23
set_transport() (*waterbutler.core.streams.StringStream method*), 27
SettingsDict (*class in waterbutler.settings*), 17
sign_data() (*in module waterbutler.core.signing*), 21
sign_message() (*waterbutler.core.signing.Signer method*), 21
sign_payload() (*waterbutler.core.signing.Signer method*), 21
Signer (*class in waterbutler.core.signing*), 21
size (*waterbutler.core.streams.BaseStream attribute*), 22
size (*waterbutler.core.streams.CutoffStream attribute*), 31
size (*waterbutler.core.streams.FileStreamReader attribute*), 25
size (*waterbutler.core.streams.FormDataStream attribute*), 31
size (*waterbutler.core.streams.MultiStream attribute*), 28
size (*waterbutler.core.streams.RequestStreamReader attribute*), 23
size (*waterbutler.core.streams.ResponseStreamReader attribute*), 22
size (*waterbutler.core.streams.StringStream attribute*), 26

SOURCE (*waterbutler.core.auth.AuthType attribute*), 19
streams (*waterbutler.core.streams.FormDataStream attribute*), 31
streams (*waterbutler.core.streams.MultiStream attribute*), 28
StringStream (*class in waterbutler.core.streams*), 26

U

unserialize_payload() (*in module waterbutler.core.signing*), 21
UtilMixin (*class in waterbutler.server.utils*), 32

V

verify_message() (*waterbutler.core.signing.Signer method*), 21
verify_payload() (*waterbutler.core.signing.Signer method*), 21

W

waterbutler (*module*), 19
waterbutler.constants (*module*), 17
waterbutler.core (*module*), 21
waterbutler.core.auth (*module*), 19
waterbutler.core.logging (*module*), 19
waterbutler.core.metrics (*module*), 19
waterbutler.core.signing (*module*), 21
waterbutler.providers (*module*), 31
waterbutler.server (*module*), 32
waterbutler.server.settings (*module*), 31
waterbutler.server.utils (*module*), 31
waterbutler.settings (*module*), 17
waterbutler.sizes (*module*), 18
waterbutler.version (*module*), 18
write() (*waterbutler.core.streams.HashStreamWriter method*), 26
write_stream() (*waterbutler.server.utils.UtilMixin method*), 32