gcs-client Documentation

Release 0.2.2

Gorka Eguileor

November 26, 2016

Contents

Contents:

CHAPTER 1

GCS-Client

Google Cloud Storage Python Client

- Apache 2.0 License
- Documentation: https://gcs-client.readthedocs.org.

The idea is to create a client with similar functionality to Google's appengine-gcs-client but intended for applications running from outside Google's AppEngine.

Cloud Storage documentation can be found at Google

1.1 Features

For now only basic functionality is available:

- Creating buckets
- Deleting buckets
- Listing buckets in a project
- Getting default bucket for a project
- Getting bucket attributes
- Listing objects in a bucket
- Getting objects attributes
- Deleting objects
- Reading object contents
- Writing an object
- Configurable retries with Truncated Exponential Backoff

1.2 Installation

To install all you need to do is run:

```
$ pip install --upgrade gcs-client
```

1.3 Usage Example

To use gcs-client in a project you will need to have Credentials to access intended Google Cloud Storage.

Credentials are generated in Google Developers Console in the Credentials section of the API Manager of the project. Recommended credentials file is JSON.

Once you have the credentials you can start using gcs_client to access your project:

```
import gcs_client

credentials = gcs_client.Credentials('private_key.json')
project = gcs_client.Project('project_name', credentials)

# Print buckets in the project
buckets = project.list()
print 'Buckets:\n\t-', '\n\t- '.join(map(str, buckets))

# Print some information from first bucket
bucket = buckets[0]
print 'Bucket %s is located in %s with storage class %s' % (bucket, bucket.location,
                                                            bucket.storageClass)

# List the objects in the bucket
objects = bucket.list()
if not objects:
    print 'There are no objects, creating one'
    filename = '/tmp/my_file.txt'
    with bucket.open(filename, 'w') as f:
        f.write('this is a test file\n' * 100)
    objects = [gcs_client.Object(bucket.name, filename, credentials=credentials)]

if objects:
    print '\t', '\n\t'.join(map(lambda o: o.name + ' has %s bytes' % o.size, objects))
    # Read the contents from the first file
    with objects[0].open() as obj:
        print 'Contents of file %s are:\n' % obj.name, obj.read()
else:
    print 'There are no objects, nothing to do'
```

More examples can be found in the documentation, in the Usage section.

1.4 Reporting an issue

If you've found an issue with gcs-client here's how you can report the problem:

- Preferred method is filing a bug on GitHub:
  1. Go to project's issue tracker on GitHub
  2. Search for existing issues using the search field at the top of the page
  3. File a new issue with information on the problem
  4. Thanks for helping make gcs-client better
- If you don't have a GitHub account and don't wish to create one you can just drop me an email.

CHAPTER 2

Installation

At the command line:

```
$ easy_install gcs-client
```

If you have pip installed:

```
$ pip install --upgrade gcs-client
```

It is good practice to use virtual environments, be it using virtualenv or virtualenvwrapper.

CHAPTER 3

Usage

To use gcs-client in a project you will need to have Credentials to access intended Google Cloud Storage.

Credentials are generated in Google Developers Console in the Credentials section of the API Manager of the project. Recommended credentials file is JSON.

Once you have the credentials you can start using gcs_client to access your project.

3.1 Listing buckets

```
import gcs_client

credentials_file = 'private_key.json'
project_name = 'project_name'

credentials = gcs_client.Credentials(credentials_file)
project = gcs_client.Project(project_name, credentials)

buckets = project.list()
print 'Buckets:\n\t- ','\n\t- '.join(map(str, buckets))
```

3.2 Creating a bucket

```
import gcs_client

credentials_file = 'private_key.json'
project_name = 'project_name'

credentials = gcs_client.Credentials(credentials_file)
project = gcs_client.Project(project_name, credentials)

bucket = project.create_bucket('my_new_bucket', location='EU')
print 'Bucket %s is located in %s with storage class %s' % (bucket, bucket.location,
                                                           bucket.storageClass)
```

3.3 Deleting a bucket

```
import gcs_client

credentials_file = 'private_key.json'
project_name = 'project_name'

credentials = gcs_client.Credentials(credentials_file)
project = gcs_client.Project(project_name, credentials)

buckets = project.list()
# Delete one bucket but never the default bucket
default_bucket = project.default_bucket_name
filtered = filter(lambda b: b.name != default_bucket, buckets)
if filtered:
    buckets[0].delete()
```

3.4 Listing all objects

```
import gcs_client

credentials_file = 'private_key.json'
project_name = 'project_name'

credentials = gcs_client.Credentials(credentials_file)
project = gcs_client.Project(project_name, credentials)

buckets = project.list()
objects = buckets[0].list()

print 'Contents of bucket %s:' % bucket
if objects:
    print '\t','\n\t'.join(map(lambda o: o.name + ' has %s bytes' % o.size, objects))
else:
    print '\tThere are no objects'
```

3.5 Listing objects with a prefix

```
import gcs_client

credentials = gcs_client.Credentials('private_key.json')
bucket = gcs_client.Bucket('bucket_name', credentials)
directory = 'var/log'
objects = bucket.list(directory)

print 'Contents of %s/%s:' % (bucket.name, directory)
if objects:
    print '\t','\n\t'.join(map(lambda o: o.name + ' has %s bytes' % o.size, objects))
else:
    print '\tThere are no objects'
```

3.6 List contents of a bucket as a directory

We can list a bucket as if it were a directory by passing `delimiter` optional argument on the `list` call.

Following example implements `tree` command for a bucket.

This is just for demonstration purposes, since it is not efficient because for each "directory" we find we make another request to the server to list its contents. It would be more efficient to request all the objects in one go and then rebuild the tree locally.

```
import gcs_client

def print_obj(obj, i, last):
    if isinstance(obj, gcs_client.Prefix):
        name = obj.prefix.split('/')[-2]
    else:
        name = obj.name.split('/')[-1]
    print ('    ' * i) + ('--' if last else '--'), name

def tree(objs, indent=0):
    if indent == 0:
        print '.'
    for i in range(len(objs)):
        obj = objs[i]
        print_obj(obj, indent, i == len(objs) - 1)
        if isinstance(obj, gcs_client.Prefix):
            tree(obj.list(), indent+1)

credentials = gcs_client.Credentials('private_key.json')
bucket = gcs_client.Bucket('bucket_name', credentials)

print '$ tree', bucket.name
tree(bucket.list(delimiter='/'))
```

3.7 Deleting objects

```
import gcs_client

credentials_file = 'private_key.json'
project_name = 'project_name'

credentials = gcs_client.Credentials(credentials_file)
project = gcs_client.Project(project_name, credentials)

bucket = project.list()[0]
objects = bucket.list()

if objects:
    obj = objects[0]
    print 'Deleting object %s' % obj
    obj.delete()
```

3.8 Reading objects

```
import gcs_client

credentials_file = 'private_key.json'
project_name = 'project_name'

credentials = gcs_client.Credentials(credentials_file)
project = gcs_client.Project(project_name, credentials)

buckets = project.list()
objects = buckets[0].list()

if objects:
    with objects[0].open() as obj:
        print 'Contents of file %s are:\n' % obj.name, obj.read()
```

3.9 Reading objects in big chunks

```
import gcs_client

credentials = gcs_client.Credentials('private_key.json')
bucket = gcs_client.Bucket('bucket_name', credentials)

chunksize = 4 * 1024 * 1024

with bucket.open('my_file', 'r', chunksize=chunksize) as obj:
    print 'Contents of file %s are:\n' % obj.name, obj.read()
```

3.10 Writing objects

```
import gcs_client

credentials_file = 'private_key.json'
project_name = 'project_name'

credentials = gcs_client.Credentials(credentials_file)
project = gcs_client.Project(project_name, credentials)

bucket = project.list()[0]

with bucket.open('new_file.txt', 'w') as obj:
    obj.write('Hello world\n')

with bucket.open('new_file.txt') as obj:
    print obj.read()
```

3.11 Changing default retry configuration

All operations use retries with Truncated Exponential Backoff by default, but we can change default configuration.

```
import gcs_client

# Set default retry configuration using a RetryParams instance
new_retry_cfg = gcs_client.RetryParams(max_retries=10, initial_delay=0.5, max_backoff=8,
                                       randomize=False)
gcs_client.RetryParams.set_default(new_retry_cfg)

# Set default retry configuration via params
gcs_client.RetryParams.set_default(max_retries=10, initial_delay=0.5, max_backoff=8,
                                   randomize=False)
```

3.12 Disabling default retries

We may want to disable all retries for all instances that are using default retry configuration. Those that are using specific instance configurations will continue doing so.

```
import gcs_client

# Disable retry configuration
gcs_client.RetryParams.set_default(0)
```

3.13 Per instance retry configuration

We can set specific retry configuration for an instance. Important to notice that listed objects will inherit retry configuration from the object that did the listing.

```
import gcs_client

credentials_file = 'private_key.json'
project_name = 'project_name'

credentials = gcs_client.Credentials(credentials_file)
project = gcs_client.Project(project_name, credentials)

bucket = project.list()[0]
# Set bucket retry configuration
bucket.retry_params = gcs_client.RetryParams(max_retries=10, initial_delay=0.5, max_backoff=8,
                                             randomize=False)

# Disable retries on the bucket
bucket.retry_params = None
```

CHAPTER 4

gcs_client package

Client Library for Google Cloud Storage.

4.1 Submodules

4.1.1 gcs_client.bucket module

class gcs_client.bucket.**Bucket**(*name=None*, *credentials=None*, *retry_params=None*)

Bases: gcs_client.base.Fillable, gcs_client.base.Listable

GCS Bucket Object representation.

Variables

- **kind** (*string*) – The kind of item this is. For buckets, this is always storage#bucket.
- **name** (*string*) – The name of the bucket.
- **timeCreated** (*string*) – The creation time of the bucket in RFC 3339 format.
- **updated** (*string*) – The modification time of the bucket in RFC 3339 format.
- **id** (*string*) – The ID of the bucket.
- **metageneration** (*long*) – The metadata generation of this bucket.
- **location** (*string*) – The location of the bucket. Object data for objects in the bucket resides in physical storage within this region. Defaults to US. See the developer's guide for the authoritative list: https://cloud.google.com/storage/docs/bucket-locations
- **owner** (*dict*) – The owner of the object. This will always be the uploader of the object. Contains entity and entityId keys.
- **etag** (*string*) – HTTP 1.1 Entity tag for the object.
- **projectNumber** (*long*) – The project number of the project the bucket belongs to.
- **selfLink** (*string*) – The link to this object.
- **storageClass** (*string*) – Storage class of the object.

Initialize a Bucket object.

Parameters

- **name** (*String*) – Name of the bucket to use.

- **credentials** (*gcs_client.Credentials*) – A credentials object to authorize the connection.
- **retry_params** (*RetryParams or NoneType*) – Retry configuration used for communications with GCS. If None is passed default retries will be used.

delete(**args*, ***kwargs*)
Permanently deletes an empty bucket from a Project.

The authenticated user in credentials must be a member of the project's team as an editor or owner.

Parameters

- **if_metageneration_match** (*long*) – If set, only deletes the bucket if its metagen-eration matches this value.
- **if_metageneration_not_match** (*long*) – If set, only deletes the bucket if its metageneration does not match this value.

Returns None

exists(**args*, ***kwargs*)
Check if exists in GCS server.

list(*prefix=None*, *maxResults=None*, *versions=None*, *delimiter=None*, *projection=None*, *pageToken=None*)
List Objects matching the criteria contained in the Bucket.

In conjunction with the prefix filter, the use of the delimiter parameter allows the list method to operate like a directory listing, despite the object namespace being flat. For example, if delimiter were set to "/", then listing objects from a bucket that contains the objects "a/b", "a/c", "d", "e", "e/f" would return objects "d" and "e", and prefixes "a/" and "e/".

The authenticated user must have READER permissions on the bucket.

Object list operations are eventually consistent. This means that if you upload an object to a bucket and then immediately perform a list operation on the bucket in which the object is stored, the uploaded object might not immediately appear in the returned list of objects. However, you can always immediately download a newly-created object and get its ACLs because object uploads are strongly consistent.

Parameters

- **prefix** (*String*) – Filter results to objects whose names begin with this prefix.
- **maxResults** (*Unsigned integer*) – Maximum number of items plus prefixes to return. As duplicate prefixes are omitted, fewer total results may be returned than requested. The default value of this parameter is 1,000 items.
- **versions** (*bool*) – If True, lists all versions of an object as distinct results. The default is False.
- **delimiter** (*String*) – Returns results in a directory-like mode. Objects whose names, aside from the prefix, do not contain delimiter with be returned as Object instances. Objects whose names, aside from the prefix, contain delimiter will be returned as Prefix instances. Duplicate prefixes are omitted.
- **projection** (*String*) – Set of properties to return. Defaults to noAcl. Acceptable values are:

  "full": Include all properties. "noAcl": Omit the acl property.

- **pageToken** (*String*) – A previously-returned page token representing part of the larger set of results to view. The pageToken is an encoded field representing the name and generation of the last object in the returned list. In a subsequent request using the

pageToken, items that come after the pageToken are shown (up to maxResults). Object list operations are eventually consistent. In addition, if you start a listing and then create an object in the bucket before using a pageToken to continue listing, you will not see the new object in subsequent listing results if it is in part of the object namespace already listed.

Returns List of objects and prefixes that match the criteria.

Return type List of gcs_client.Object and gcs_client.Prefix.

open(*name*, *mode='r'*, *generation=None*, *chunksize=None*)
Open an object from the Bucket.

Parameters

- **name** (*String*) – Name of the file to open.
- **mode** (*String*) – Mode to open the file with, 'r' for read and 'w' for writing are only supported formats. Default is 'r' if this argument is not provided.
- **generation** (*long*) – If present, selects a specific revision of this object (as opposed to the latest version, the default).
- **chunksize** (*int*) – Size in bytes of the payload to send/receive to/from GCS. Default is gcs_client.DEFAULT_BLOCK_SIZE

credentials
Credentials used to connect to GCS server.

kind = 'storage#buckets'

retry_params
Get retry configuration used by this instance for accessing GCS.

4.1.2 gcs_client.constants module

gcs_client.constants.**ACL_AUTH_READ = 'authenticatedRead'**
Project team/Object owners get OWNER access, and allAuthenticatedUsers get READER access.

gcs_client.constants.**ACL_OWNER_FULL = 'bucketOwnerFullControl'**
Object owner gets OWNER access, and project team owners get OWNER access.

gcs_client.constants.**ACL_OWNER_READ = 'bucketOwnerRead'**
Object owner gets OWNER access, and project team owners get READER access.

gcs_client.constants.**ACL_PRIVATE = 'private'**
Project team/Object owners get OWNER access.

gcs_client.constants.**ACL_PROJECT_PRIVATE = 'projectPrivate'**
Object owner gets OWNER access, and project team members get access according to their roles.

gcs_client.constants.**ACL_PUBLIC_R = 'publicRead'**
Project team owners get OWNER access, and allUsers get READER access.

gcs_client.constants.**ACL_PUBLIC_RW = 'publicRead'**
Project team/Object owners get OWNER access, and allUsers get WRITER access.

gcs_client.constants.**PROJECTION_FULL = 'full'**
Full projection: Include all properties....

gcs_client.constants.**PROJECTION_SIMPLE = 'noAcl'**
Simple projection: Exclude the ACL property.

gcs_client.constants.**SCOPE_CLOUD = 'CLOUD'**
Cloud permissions: For buckets, they have the predefined project-private ACL applied when they are created. Buckets are always owned by the project-owners group. For objects, they have the predefined project-private ACL applied when they are uploaded. Objects are always owned by the original requester who uploaded the object.

gcs_client.constants.**SCOPE_OWNER = 'OWNER'**
Owner permissions: For buckets gives a user READER and WRITER permissions on the bucket. It also lets a user read and write bucket metadata, including ACLs. For objects, gives a user READER access. It also lets a user read and write object metadata, including ACLs.

gcs_client.constants.**SCOPE_READER = 'READER'**
Reader permissions lets a user download an object's data and list a bucket's contents.

gcs_client.constants.**SCOPE_WRITER = 'WRITER'**
Writer permissions: For buckets lets a user list, create, overwrite, and delete objects in a bucket. You cannot apply this permission to objects.

gcs_client.constants.**STORAGE_DURABLE = 'DURABLE_REDUCED_AVAILABILITY'**
Durable Reduced Availability (DRA) class: Lower availability than Standard Storage and lower cost per GB stored. Use cases: Applications that are particularly cost-sensitive, or for which some unavailability is acceptable such as batch jobs and some types of data backup.

gcs_client.constants.**STORAGE_NEARLINE = 'NEARLINE'**
Cloud Storage Nearline class: Slightly lower availability and slightly higher latency (time to first byte is typically 2 - 5 seconds) than Standard Storage but with a lower cost. Use cases: Data you do not expect to access frequently (i.e., no more than once per month). Typically this is backup data for disaster recovery, or so called "cold" storage that is archived and may or may not be needed at some future time.

gcs_client.constants.**STORAGE_STANDARD = 'STANDARD'**
Standard Storage class: High availability, low latency (time to first byte is typically tens of milliseconds). Use cases: Storing data that requires low latency access or data that is frequently accessed ("hot" objects), such as serving website content, interactive workloads, or gaming and mobile applications.

4.1.3 gcs_client.credentials module

class gcs_client.credentials.**Credentials**(*key_file_name*, *email=None*, *scope='OWNER'*)
Bases: oauth2client.client.SignedJwtAssertionCredentials

GCS Credentials used to access servers.

Initialize credentials used for all GCS operations.

Create OAuth 2.0 credentials to access GCS from a JSON file or a P12 and email address.

Since this library is meant to work outside of Google App Engine and Google Compute Engine, you must obtain these credential files in the Google Developers Console. To generate service-account credentials, or to view the public credentials that you've already generated, do the following:

1. Open the Credentials page.
2. To set up a new service account, do the following:
   (a) Click Add credentials > Service account.
   (b) Choose whether to download the service account's public/private key as a JSON file (preferred) or standard P12 file.

Your new public/private key pair is generated and downloaded to your machine; it serves as the only copy of this key. You are responsible for storing it securely.

You can return to the Developers Console at any time to view the client ID, email address, and public key fingerprints, or to generate additional public/private key pairs. For more details about service account credentials in the Developers Console, see Service accounts in the Developers Console help file.

Parameters

- **key_file_name** (*String*) – Name of the file with the credentials to use.
- **email** (*String*) – Service account's Email address to use with P12 file. When using JSON files this argument will be ignored.
- **scope** (*String*) – Scopes that the credentials should be granted access to. Value must be one of Credentials.scope_urls.keys()

apply(*headers*)

Add the authorization to the headers.

Args: headers: dict, the headers to add the Authorization header to.

authorize(*http*)

Authorize an httplib2.Http instance with these credentials.

The modified http.request method will add authentication headers to each request and will refresh access_tokens when a 401 is received on a request. In addition the http.request method has a credentials property, http.request.credentials, which is the Credentials object that authorized it.

Args:

http: An instance of httplib2.Http or something that acts like it.

Returns: A modified instance of http that was passed in.

Example:

```
h = httplib2.Http()
h = credentials.authorize(h)
```

You can't create a new OAuth subclass of httplib2.Authentication because it never gets passed the absolute URI, which is needed for signing. So instead we have to overload 'request' with a closure that adds in the Authorization header and then calls the original version of 'request()'.

create_scoped(*scopes*)

Create a Credentials object for the given scopes.

The Credentials type is preserved.

create_scoped_required()

Whether this Credentials object is scopeless.

create_scoped(scopes) method needs to be called in order to create a Credentials object for API calls.

from_json(*s*)

from_stream(*credential_filename*)

Create a Credentials object by reading information from a file.

It returns an object of type GoogleCredentials.

Args:

credential_filename: the path to the file from where the credentials are to be read

Raises:

ApplicationDefaultCredentialsError: raised when the credentials fail to be retrieved.

get_access_token(*http=None*)
Return the access token and its expiration information.

If the token does not exist, get one. If the token expired, refresh it.

get_application_default()
Get the Application Default Credentials for the current environment.

Raises:

ApplicationDefaultCredentialsError: raised when the credentials fail to be retrieved.

has_scopes(*scopes*)
Verify that the credentials are authorized for the given scopes.

Returns True if the credentials authorized scopes contain all of the scopes given.

Args: scopes: list or string, the scopes to check.

Notes: There are cases where the credentials are unaware of which scopes are authorized. Notably, credentials obtained and stored before this code was added will not have scopes, AccessTokenCredentials do not have scopes. In both cases, you can use refresh_scopes() to obtain the canonical set of scopes.

new_from_json(*s*)
Utility class method to instantiate a Credentials subclass from JSON.

Expects the JSON string to have been produced by to_json().

Args: s: string or bytes, JSON from to_json().

Returns: An instance of the subclass of Credentials that was serialized with to_json().

refresh(*http*)
Forces a refresh of the access_token.

Args:

http: httplib2.Http, an http object to be used to make the refresh request.

retrieve_scopes(*http*)
Retrieves the canonical list of scopes for this access token.

Gets the scopes from the OAuth2 provider.

Args:

http: httplib2.Http, an http object to be used to make the refresh request.

Returns: A set of strings containing the canonical list of scopes.

revoke(*http*)
Revokes a refresh_token and makes the credentials void.

Args:

http: httplib2.Http, an http object to be used to make the revoke request.

set_store(*store*)
Set the Storage for the credential.

Args:

store: Storage, an implementation of Storage object. This is needed to store the latest access_token if it has expired and been refreshed. This implementation uses locking to check for updates before updating the access_token.

to_json()

MAX_TOKEN_LIFETIME_SECS = 3600

NON_SERIALIZED_MEMBERS = ['store']

access_token_expired
True if the credential is expired or invalid.

If the token_expiry isn't set, we assume the token doesn't expire.

authorization
Authorization header value for GCS requests.

common_url = 'https://www.googleapis.com/auth/'

scope_urls = {'OWNER': 'devstorage.full_control', 'WRITER': 'devstorage.read_write', 'CLOUD': 'cloud-platform',

serialization_data
Get the fields and values identifying the current credentials.

4.1.4 gcs_client.errors module

exception gcs_client.errors.**BadGateway**(*message=None*, *code=None*)
Bases: *gcs_client.errors.Transient*

args

code = 502

message

exception gcs_client.errors.**BadRequest**(*message=None*, *code=None*)
Bases: *gcs_client.errors.Fatal*

args

code = 400

message

exception gcs_client.errors.**Credentials**
Bases: *gcs_client.errors.Error*

Credentials errors.

args

message

exception gcs_client.errors.**Error**
Bases: exceptions.Exception

Base error for all gcs_client operations.

args

message

exception gcs_client.errors.**Fatal**(*message=None*, *code=None*)
Bases: *gcs_client.errors.Http*

Fatal HTTP exceptions.

args

code = None

message

exception gcs_client.errors.**Forbidden**(*message=None*, *code=None*)
Bases: *gcs_client.errors.Fatal*

args

code = 403

message

exception gcs_client.errors.**GatewayTimeout**(*message=None*, *code=None*)
Bases: *gcs_client.errors.Transient*

args

code = 504

message

exception gcs_client.errors.**Http**(*message=None*, *code=None*)
Bases: *gcs_client.errors.Error*

HTTP specific errors.

args

code = None

message

exception gcs_client.errors.**InternalServer**(*message=None*, *code=None*)
Bases: *gcs_client.errors.Transient*

args

code = 500

message

exception gcs_client.errors.**InvalidRange**(*message=None*, *code=None*)
Bases: *gcs_client.errors.Fatal*

args

code = 416

message

exception gcs_client.errors.**NotFound**(*message=None*, *code=None*)
Bases: *gcs_client.errors.Fatal*

args

code = 404

message

exception gcs_client.errors.**RequestTimeout**(*message=None*, *code=None*)
Bases: *gcs_client.errors.Transient*

args

code = 408

message

exception gcs_client.errors.**ServiceUnavailable**(*message=None*, *code=None*)
Bases: *gcs_client.errors.Transient*

args

code = 503

message

exception gcs_client.errors.**TooManyRequests**(*message=None*, *code=None*)
Bases: *gcs_client.errors.Transient*

args

code = 429

message

exception gcs_client.errors.**Transient**(*message=None*, *code=None*)
Bases: *gcs_client.errors.Http*

Transient HTTP exceptions.

args

code = None

message

exception gcs_client.errors.**Unauthorized**(*message=None*, *code=None*)
Bases: *gcs_client.errors.Fatal*

args

code = 401

message

gcs_client.errors.**error_class**
alias of *Transient*

gcs_client.errors.**new_class**
alias of *GatewayTimeout*

gcs_client.errors.**create_http_exception**(*status_code*, *message=None*)
Create an http exception.

Create an Http exception instance as specific as possible.

For status codes that have specific exceptions, like with 408 (RequestTimeout class), those will be returned, but for those that we don't have one we will return a generic Http error with the right status code.

Parameters

- **status_code** (*int or string*) – Status code of the http error
- **message** (*str*) – Detailed message for the error

Returns Http exception instance as specific as possible

Return type Http or subclass

4.1.5 gcs_client.gcs_object module

class gcs_client.gcs_object.**Object**(*bucket=None*, *name=None*, *generation=None*, *credentials=None*, *retry_params=None*, *chunksize=None*)
Bases: gcs_client.base.Fillable

GCS Stored Object Object representation.

Variables

- **bucket** (*string*) – The name of the bucket containing this object.
- **contentType** (*string*) – Content-Type of the object data.
- **crc32c** (*string*) – CRC32c checksum, as described in RFC 4960, Appendix B; encoded using base64 in big-endian byte order.
- **etag** (*string*) – HTTP 1.1 Entity tag for the object.
- **generation** (*long*) – The content generation of this object. Used for object versioning.
- **id** (*string*) – The ID of the object.
- **kind** (*string*) – The kind of item this is. For objects, this is always storage#object.
- **md5Hash** (*string*) – MD5 hash of the data; encoded using base64.
- **mediaLink** (*string*) – Media download link.
- **metadata** (*dict*) – User-provided metadata, in key/value pairs.
- **metageneration** (*long*) – The version of the metadata for this object at this generation. Used for preconditions and for detecting changes in metadata. A metageneration number is only meaningful in the context of a particular generation of a particular object.
- **name** (*string*) – The name of this object.
- **owner** (*dict*) – The owner of the object. This will always be the uploader of the object. Contains entity and entityId keys.
- **selfLink** (*string*) – The link to this object.
- **size** (*unsigned long*) – Content-Length of the data in bytes.
- **storageClass** (*string*) – Storage class of the object.
- **timeCreated** (*string*) – The creation time of the object in RFC 3339 format.
- **timeDeleted** (*string*) – The deletion time of the object in RFC 3339 format. Will be None if this version of the object has not been deleted.
- **updated** (*string*) – The modification time of the object metadata in RFC 3339 format.

Initialize an Object object.

Parameters

- **bucket** (*String*) – Name of the bucket to use.
- **name** (*String*) – Name of the object.
- **generation** (*long*) – If present, selects a specific revision of this object (as opposed to the latest version, the default).
- **credentials** (Credentials) – A credentials object to authorize the connection.
- **retry_params** (*RetryParams or NoneType*) – Retry configuration used for communications with GCS. If None is passed default retries will be used.
- **chunksize** (*int*) – Size in bytes of the payload to send/receive to/from GCS. Default is gcs_client.DEFAULT_BLOCK_SIZE

delete(**args*, ***kwargs*)

Deletes an object and its metadata.

Deletions are permanent if versioning is not enabled for the bucket, or if the generation parameter is used.

The authenticated user in the credentials must have WRITER permissions on the bucket.

Parameters

- **generation** (*long*) – If present, permanently deletes a specific revision of this object (as opposed to the latest version, the default).
- **if_generation_match** (*long*) – Makes the operation conditional on whether the object's current generation matches the given value.
- **if_generation_not_match** (*long*) – Makes the operation conditional on whether the object's current generation does not match the given value.
- **if_metageneration_match** (*long*) – Makes the operation conditional on whether the object's current metageneration matches the given value.
- **if_metageneration_not_match** (*long*) – Makes the operation conditional on whether the object's current metageneration does not match the given value.

Returns None

exists(**args*, ***kwargs*)
Check if exists in GCS server.

open(**args*, ***kwargs*)
Open this object.

Parameters

- **mode** (*String*) – Mode to open the file with, 'r' for read and 'w' for writing are only supported formats. Default is 'r' if this argument is not provided.
- **chunksize** (*int*) – Size in bytes of the payload to send/receive to/from GCS. Default chunksize is the one defined on object's initialization.

credentials
Credentials used to connect to GCS server.

kind = 'storage#objects'

metadata = {}

retry_params
Get retry configuration used by this instance for accessing GCS.

timeDeleted = None

class gcs_client.gcs_object.**GCSObjFile**(*bucket*, *name*, *credentials*, *mode='r'*, *chunksize=None*, *retry_params=None*, *generation=None*)

Bases: object

Reader/Writer for GCS Objects.

Supports basic functionality:

- Read
- Write
- Close
- Seek
- Tell

Instances support context manager behavior.

Initialize reader/writer of GCS object.

On initialization connection to GCS will be tested. For reading it'll confirm the existence of the object in the bucket and for writing it'll create the object (it won't send any content).

Parameters

- **bucket** (*String*) – Name of the bucket to use.
- **name** (*String*) – Name of the object.
- **credentials** (*gcs_client.Credentials*) – A credentials object to authorize the connection.
- **mode** (*String*) – Mode to open the file with, 'r' for read and 'w' for writing are only supported formats. Default is 'r' if this argument is not provided.
- **chunksize** (*int*) – Size in bytes of the payload to send/receive to/from GCS. Default is gcs_client.DEFAULT_BLOCK_SIZE
- **retry_params** (*RetryParams or NoneType*) – Retry configuration used for communications with GCS. If None is passed default retries will be used.
- **generation** (*long*) – If present, selects a specific revision of this object (as opposed to the latest version, the default).

close()

Close the file.

A closed file cannot be read or written any more. Any operation which requires that the file be open will raise an error after the file has been closed. Calling close() more than once is allowed.

read(*size=None*)

Read data from the file.

Read at most size bytes from the file (less if the read hits EOF before obtaining size bytes). If the size argument is None, read all data until EOF is reached.

The bytes are returned as a bytes object. An empty string is returned when EOF is encountered immediately.

Note that this method may make multiple requests to GCS service in an effort to acquire as close to size bytes as possible.

Parameters **size** (*int*) – Number of bytes to read.

Returns Bytes with read data from GCS.

Return type bytes

seek(*offset*, *whence=0*)

Set the file's current position, like stdio's fseek().

Note that only files open for reading are seekable.

Parameters

- **offset** (*int*) – Offset to move the file cursor.
- **whence** (*int*) – How to interpret the offset, defaults to os.SEEK_SET (0) -absolute file positioning- other values are os.SEEK_CUR (1) -seek relative to the current position- and os.SEEK_END (2) -seek relative to the file's end-.

Returns None

tell()

Return file's current position from the beginning of the file.

write(*data*)
Write a string to the file.

Due to buffering, the string may not actually show up in the file until we close the file or enough data to send another chunk has been buffered.

Parameters **data** (*String*) – Data to write to the object.

Returns None

4.1.6 gcs_client.prefix module

class gcs_client.prefix.**Prefix**(*name*, *prefix*, *delimiter=None*, *credentials=None*, *retry_params=None*)
Bases: gcs_client.base.Listable

GCS Prefix Object representation.

Represents prefixes returned from listing objects inside a bucket using a delimiter.

A Prefix object can be thought of as a directory in a filesystem.

Variables

- ***kind*** (*string*) – The kind of item this is. For buckets, this is always storage#prefix.
- **name** (*string*) – Bucket name that this prefix belongs to.
- ***prefix*** (*string*) – Prefix name (like the full path of a directory).
- **delimiter** (*string*) – Delimiter used to generate this prefix.

Initialize a prefix representation.

Parameters

- **name** (*String*) – Name of the bucket this prefix belongs to.
- **prefix** (*String*) – Prefix name (like the full path of a directory).
- **delimiter** (*String*) – Delimiter used on the listing
- **credentials** (*gcs_client.Credentials*) – A credentials object to authorize the connection.
- **retry_params** (*RetryParams or NoneType*) – Retry configuration used for communications with GCS. If None is passed default retries will be used.

exists(**args*, ***kwargs*)
Check if exists in GCS server.

list(*prefix=''*, *maxResults=None*, *versions=None*, *delimiter=None*, *projection=None*, *pageToken=None*)
List Objects matching the criteria contained in the Bucket.

In conjunction with the prefix filter, the use of the delimiter parameter allows the list method to operate like a directory listing, despite the object namespace being flat. For example, if delimiter were set to "/", then listing objects from a bucket that contains the objects "a/b", "a/c", "d", "e", "e/f" would return objects "d" and "e", and prefixes "a/" and "e/".

The authenticated user must have READER permissions on the bucket.

Object list operations are eventually consistent. This means that if you upload an object to a bucket and then immediately perform a list operation on the bucket in which the object is stored, the uploaded object might

not immediately appear in the returned list of objects. However, you can always immediately download a newly-created object and get its ACLs because object uploads are strongly consistent.

This differs slightly from bucket listing, because if no delimiter is provided it will use the delimiter that was used on the listing that generated the instance. Likewise for the prefix.

Parameters

- **prefix** (*String*) – Filter results in this 'directory' to objects whose names begin with this prefix. Default is to list all objects in the directory.
- **maxResults** (*Unsigned integer*) – Maximum number of items plus prefixes to return. As duplicate prefixes are omitted, fewer total results may be returned than requested. The default value of this parameter is 1,000 items.
- **versions** (*bool*) – If True, lists all versions of an object as distinct results. The default is False.
- **delimiter** (*String*) – Returns results in a directory-like mode. Objects whose names, aside from the prefix, do not contain delimiter with be returned as Object instances. Objects whose names, aside from the prefix, contain delimiter will be returned as Prefix instances. Duplicate prefixes are omitted. Default use delimiter that was used to create this prefix.
- **projection** (*String*) – Set of properties to return. Defaults to noAcl. Acceptable values are:

  "full": Include all properties. "noAcl": Omit the acl property.

- **pageToken** (*String*) – A previously-returned page token representing part of the larger set of results to view. The pageToken is an encoded field representing the name and generation of the last object in the returned list. In a subsequent request using the pageToken, items that come after the pageToken are shown (up to maxResults). Object list operations are eventually consistent. In addition, if you start a listing and then create an object in the bucket before using a pageToken to continue listing, you will not see the new object in subsequent listing results if it is in part of the object namespace already listed.

Returns List of objects and prefixes that match the criteria.

Return type List of gcs_client.Object and gcs_client.Prefix.

credentials
Credentials used to connect to GCS server.

kind = 'storage#prefix'

retry_params
Get retry configuration used by this instance for accessing GCS.

4.1.7 gcs_client.project module

class gcs_client.project.**Project**(*project_id*, *credentials=None*, *retry_params=None*)
Bases: gcs_client.base.Listable

GCS Project Object representation.

Initialize a Project object.

Parameters

- **project_id** (*String*) – Project id as listed in Google's project management https://console.developers.google.com/project.
- **credentials** (*gcs_client.Credentials*) – A credentials object to authorize the connection.
- **retry_params** (*RetryParams or NoneType*) – Retry configuration used for communications with GCS. If None is passed default retries will be used.

create_bucket (**args*, ***kwargs*)

Create a new bucket in the project.

Google Cloud Storage uses a flat namespace, so you can't create a bucket with a name that is already in use.

For more information, see Bucket Naming Guidelines: https://cloud.google.com/storage/docs/bucket-naming#requirements

The authenticated user in credentials must be a member of the project's team as an editor or owner.

Parameters

- **name** (*String*) – The name of the bucket.
- **location** (*String*) – The location of the bucket. Object data for objects in the bucket resides in physical storage within this region. Defaults to US. See the developer's guide for the authoritative list: https://cloud.google.com/storage/docs/bucket-locations
- **storage_class** (*String*) – The bucket's storage class. This defines how objects in the bucket are stored and determines the SLA and the cost of storage. Value must be one of gcs_client.constants.STORAGE_*, and they include STANDARD, NEARLINE and DURABLE_REDUCED_AVAILABILITY. Defaults to gcs_client.constants.STORAGE_NEARLINE.
- **predefined_acl** (*String*) – Apply a predefined set of access controls to this bucket. Acceptable values from gcs_client.ACL_* are ACL_AUTH_READ, ACL_PRIVATE, ACL_PROJECT_PRIVATE, ACL_PUBLIC_R, and ACL_PUBLIC_RW
- **predefined_default_obj_acl** (*String*) – Apply a predefined set of default object access controls to this bucket. Acceptable values from gcs_client.constants.ACL_* are ACL_AUTH_READ, ACL_OWNER_FULL, ACL_OWNER_READ, ACL_PRIVATE, ACL_PROJECT_PRIVATE, and ACL_PUBLIC_R
- **projection** (*String*) – Set of properties to return. Defaults to PROJECTION_SIMPLE. Acceptable values from gcs_client.constants are:
  - PROJECTION_FULL ('full'): Include all properties.
  - PROJECTION_SIMPLE ('noAcl'): Omit the acl property.

Returns A new Bucket instance

Return type gcs_client.Bucket

exists (**args*, ***kwargs*)

Check if exists in GCS server.

list (*fields=None*, *maxResults=None*, *projection=None*, *prefix=None*, *pageToken=None*)

Retrieves a list of buckets for a given project.

The authenticated user in credentials must be a member of the project's team.

Bucket list operations are eventually consistent. This means that if you create a bucket and then immediately perform a list operation, the newly-created bucket will be immediately available for use, but the bucket might not immediately appear in the returned list of buckets.

Parameters

- **fields** (*list of strings*) – Limit retrieved data for each bucket to these fields.
- **maxResults** (*Unsigned integer*) – Maximum number of buckets to return.
- **projection** (*String*) – Set of properties to return. Defaults to noAcl. Acceptable values are:

  "full": Include all properties. "noAcl": Omit the acl property.
- **prefix** (*String*) – Filter results to buckets whose names begin with this prefix.
- **pageToken** (*String*) – A previously-returned page token representing part of the larger set of results to view. The pageToken is an encoded field representing the name and generation of the last bucket in the returned list. In a subsequent request using the pageToken, items that come after the pageToken are shown (up to maxResults). Bucket list operations are eventually consistent. In addition, if you start a listing and then create a new bucket before using a pageToken to continue listing, you will not see the new bucket in subsequent listing results if it is in part of the bucket namespace already listed.

Returns List of buckets that match the criteria.

Return type List of gcs_client.Bucket.

credentials
Credentials used to connect to GCS server.

default_bucket_name
Bucket name of the default bucket for the project.

retry_params
Get retry configuration used by this instance for accessing GCS.

CHAPTER 5

Contributing

Contributions are welcome, and they are greatly appreciated! Every little bit helps, and credit will always be given.

You can contribute in many ways:

5.1 Types of Contributions

5.1.1 Report Bugs

Report bugs at https://github.com/Akrog/gcs-client/issues.

If you are reporting a bug, please include:

- Your operating system name and version.
- Any details about your local setup that might be helpful in troubleshooting.
- Detailed steps to reproduce the bug.

5.1.2 Fix Bugs

Look through the GitHub issues for bugs. Anything tagged with "bug" is open to whoever wants to implement it.

5.1.3 Implement Features

Look through the GitHub issues for features. Anything tagged with "feature" is open to whoever wants to implement it.

5.1.4 Write Documentation

gcs-client could always use more documentation, whether as part of the official gcs-client docs, in docstrings, or even on the web in blog posts, articles, and such.

5.1.5 Submit Feedback

The best way to send feedback is to file an issue at https://github.com/Akrog/gcs-client/issues.

If you are proposing a feature:

- Explain in detail how it would work.
- Keep the scope as narrow as possible, to make it easier to implement.
- Remember that this is a volunteer-driven project, and that contributions are welcome :)

5.2 Get Started!

Ready to contribute? Here's how to set up *gcs-client* for local development.

1. Fork the *gcs-client* repo on GitHub.
2. Clone your fork locally:

```
$ git clone git@github.com:your_name_here/gcs-client.git
```

3. Install your local copy into a virtualenv. Assuming you have virtualenvwrapper installed, this is how you set up your fork for local development:

```
$ mkvirtualenv gcs-client
$ cd gcs-client/
$ python setup.py develop
```

4. Create a branch for local development:

```
$ git checkout -b name-of-your-bugfix-or-feature
```

   Now you can make your changes locally.

5. When you're done making changes, check that your changes pass flake8 and the tests, including testing other Python versions with tox:

```
$ flake8 gcs-client tests
$ python setup.py test
$ tox
```

   To get flake8 and tox, just pip install them into your virtualenv.

6. Commit your changes and push your branch to GitHub:

```
$ git add .
$ git commit -m "Your detailed description of your changes."
$ git push origin name-of-your-bugfix-or-feature
```

7. Submit a pull request through the GitHub website.

5.3 Pull Request Guidelines

Before you submit a pull request, check that it meets these guidelines:

1. The pull request should include tests.
2. If the pull request adds functionality, the docs should be updated. Put your new functionality into a function with a docstring, and add the feature to the list in README.rst.
3. The pull request should work for Python 2.6, 2.7, 3.3, and 3.4, and for PyPy. Check https://travis-ci.org/Akrog/gcs-client/pull_requests and make sure that the tests pass for all supported Python versions.

5.4 Tips

To run a subset of tests:

```
$ python -m unittest tests.test_gcs-client
```

CHAPTER 6

Credits

6.1 Development Lead

- Gorka Eguileor <gorka@eguileor.com>

6.2 Contributors

None yet. Why not be the first?

CHAPTER 7

History

7.1 0.2.2 (2016-11-26)

- Fix #1 - Media upload not working

7.2 0.2.1 (2016-03-26)

- Fix requirements issue

7.3 0.2.0 (2015-11-25)

- Add support for delimiter in listings using Prefix object
- Remove dependency from google-api-python-client
- Return descriptive error message on AttributeError

7.4 0.1.4 (2015-11-16)

- No changes, version bump to reload in pypi

7.5 0.1.3 (2015-11-16)

- Reading an object will return data as bytes, not a bytearray.
- Read specific versions of an object.
- Can specify chunksize on Bucket's open and Object init.
- Add exists to buckets and project.
- Fix opening files with names that needed encoding.

7.6 0.1.2 (2015-11-10)

- Fix seek method.

7.7 0.1.1 (2015-11-09)

- Fix Pypi package.

7.8 0.1.0 (2015-11-09)

- First release on PyPI.

CHAPTER 8

Indices and tables

- genindex
- search

b

c

e

g

p

A

B

C

D

E

F

G

H

I

K

L

M

N

O

P

R

S

T

U

W