python-twitter Documentation

Release 3.4.2

python-twitter@googlegroups.com

Sep 30, 2018

Contents

A Python wrapper around the Twitter API.

Author: The Python-Twitter Developers <python-twitter@googlegroups.com>

Contents:

CHAPTER 1

Installation & Testing

1.1 Installation

From PyPI

```
$ pip install python-twitter
```

From source

Install the dependencies:

- Requests
- Requests OAuthlib

Alternatively use *pip*:

```
$ pip install -r requirements.txt
```

Download the latest *python-twitter* library from: https://github.com/bear/python-twitter/

Extract the source distribution and run:

```
$ python setup.py build
$ python setup.py install
```

1.2 Testing

The following requires `pip install pytest` and `pip install pytest-cov`. Run:

```
$ make test
```

If you would like to see coverage information:

```
$ make coverage
```

1.3 Getting the code

The code is hosted at Github.

Check out the latest development version anonymously with:

```
$ git clone git://github.com/bear/python-twitter.git
$ cd python-twitter
```

CHAPTER 2

Getting Started

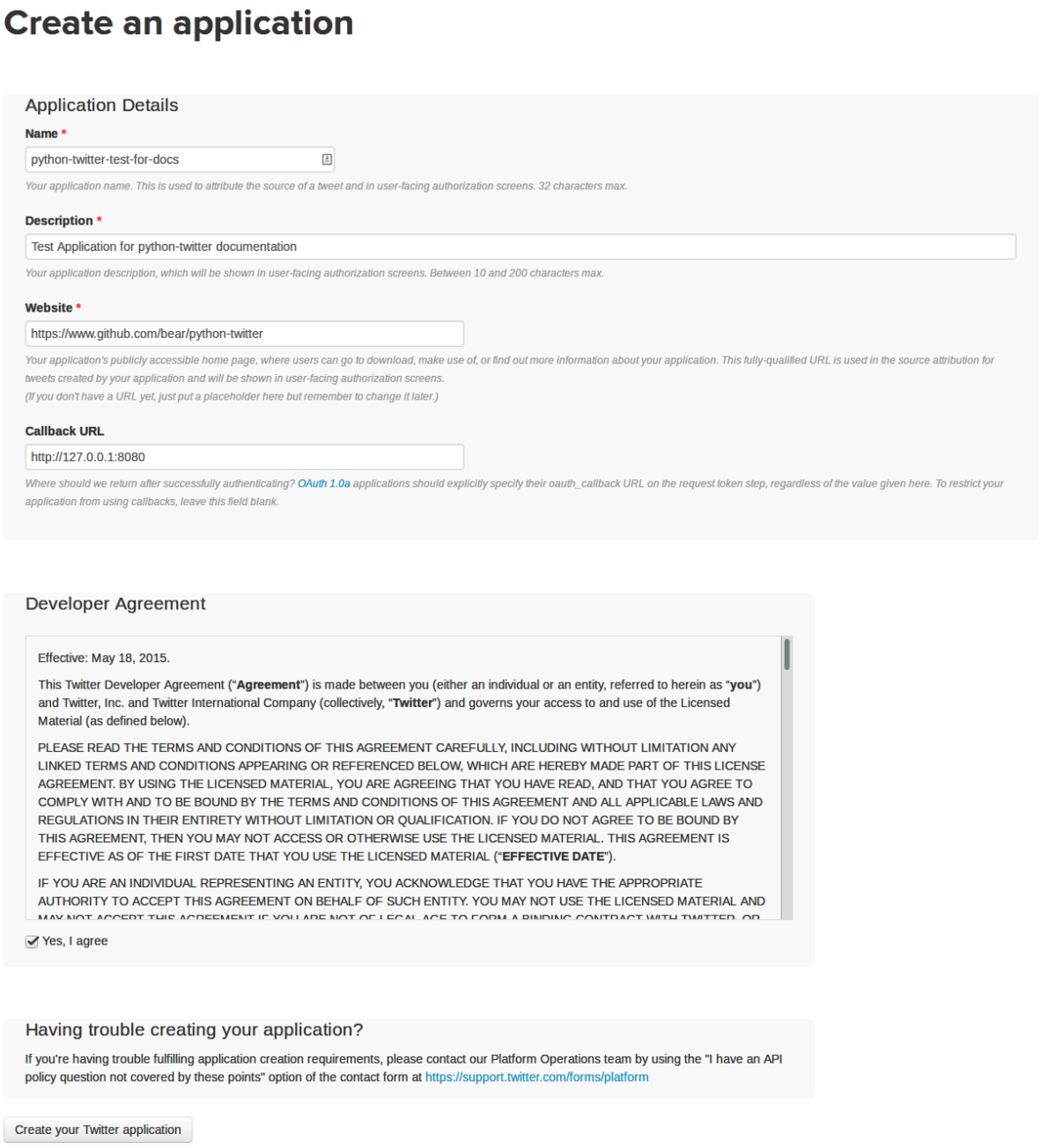
2.1 Getting your application tokens

This section is subject to changes made by Twitter and may not always be completely up-to-date. If you see something change on their end, please create a new issue on Github or submit a pull request to update it.

In order to use the python-twitter API client, you first need to acquire a set of application tokens. These will be your `consumer_key` and `consumer_secret`, which get passed to `twitter.Api()` when starting your application.

2.1.1 Create your app

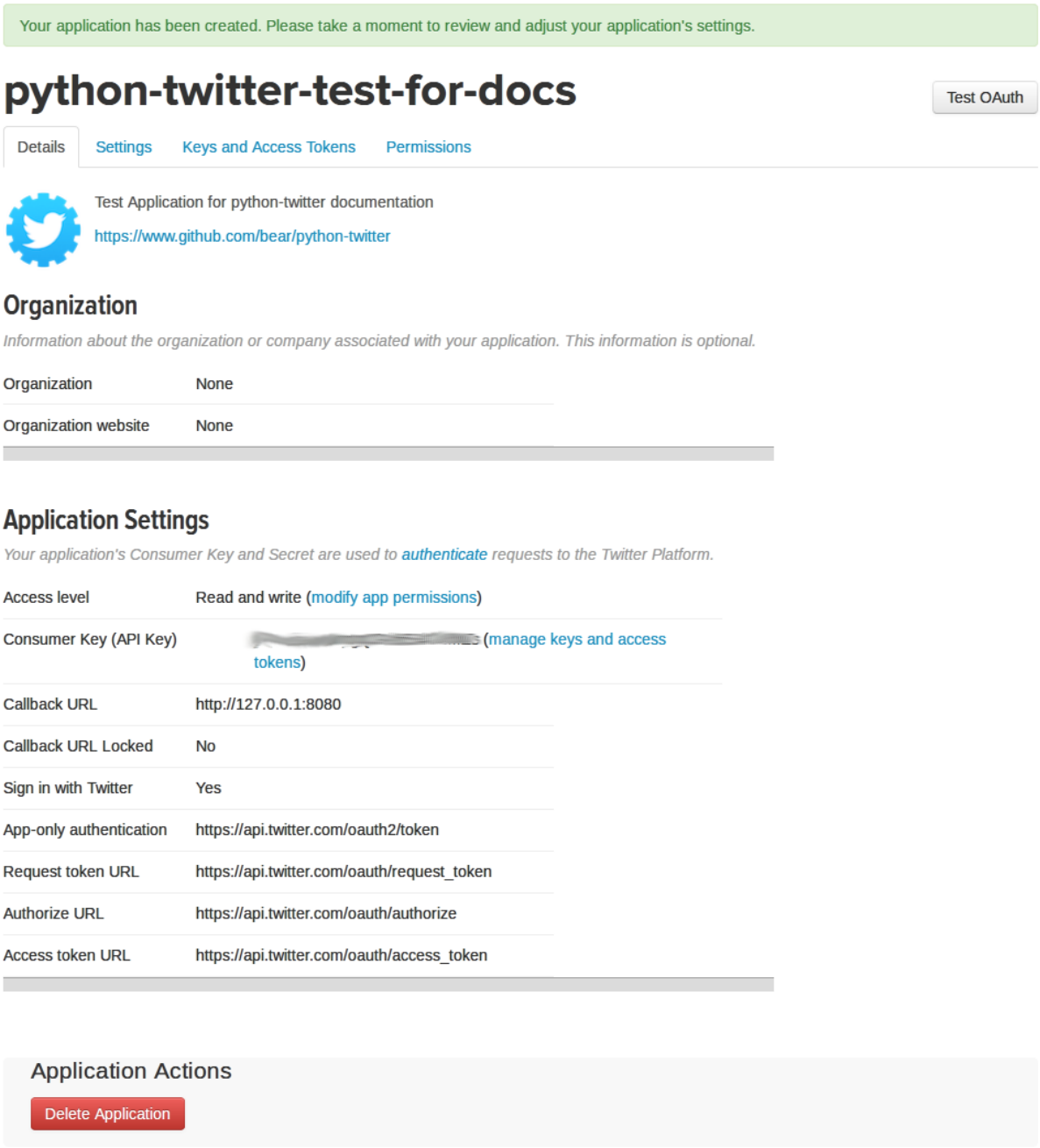
The first step in doing so is to create a Twitter App. Click the "Create New App" button and fill out the fields on the next page.



If there are any problems with the information on that page, Twitter will complain and you can fix it. (Make sure to get the name correct - it is unclear if you can change this later.) On the next screen, you'll see the application that you created and some information about it:

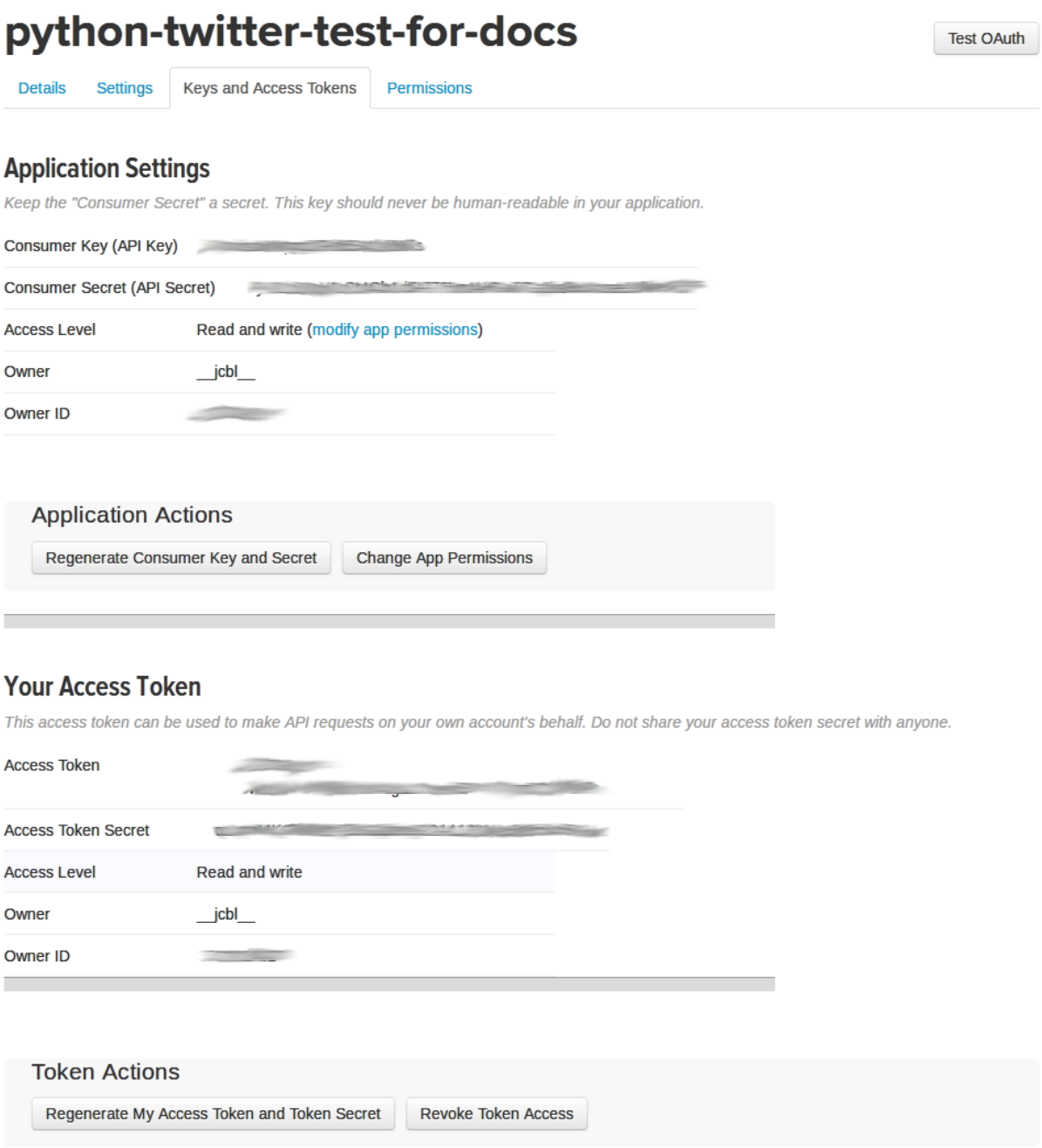
2.1.2 Your app

Once your app is created, you'll be directed to a new page showing you some information about it.



2.1.3 Your Keys

Click on the "Keys and Access Tokens" tab on the top there, just under the green notification in the image above.



At this point, you can test out your application using the keys under "Your Application Tokens". The `twitter.Api()` object can be created as follows:

```
import twitter
api = twitter.Api(consumer_key=[consumer key],
                  consumer_secret=[consumer secret],
                  access_token_key=[access token],
                  access_token_secret=[access token secret])
```

Note: Make sure to enclose your keys in quotes (ie, api = twitter.Api(consumer_key='1234567', ...) and so on) or you will receive a NameError.

If you are creating an application for end users/consumers, then you will want them to authorize you application, but

that is outside the scope of this document.

And that should be it! If you need a little more help, check out the examples on Github. If you have an open source application using python-twitter, send us a link and we'll add a link to it here.

CHAPTER 3

Contributing

3.1 Getting the code

The code is hosted at Github.

Check out the latest development version anonymously with:

```
$ git clone git://github.com/bear/python-twitter.git
$ cd python-twitter
```

The following sections assuming that you have pyenv installed and working on your computer.

To install dependencies, run:

```
$ make dev
```

This will install all of the required packages for the core library, testing, and installation.

3.2 Testing

Once you have your development environment set up, you can run:

```
$ make test
```

to ensure that all tests are currently passing before starting work. You can also check test coverage by running:

```
$ make coverage
```

Pull requests are welcome or, if you are having trouble, please open an issue on GitHub.

CHAPTER 4

Migration from v2 to v3

4.1 Changes to Existing Methods

4.1.1 `twitter.api.Api()`

- `shortner` parameter has been removed. Please see Issue #298.

4.1.2 `twitter.api.Api.CreateFavorite()`

- kwarg param has been changed to `status_id` from `id` to be consistent with other method calls and avoid shadowing builtin function `id`.

4.1.3 `twitter.api.Api.DestroyFavorite()`

- kwarg param has been changed to `status_id` from `id` to be consistent with other method calls and avoid shadowing builtin function `id`.

4.1.4 `twitter.api.Api.DestroyBlock()`

- Kwarg `id` has been changed to `user_id` in order to avoid shadowing a builtin and be more descriptive.

4.1.5 `twitter.api.Api.DestroyStatus()`

- kwarg `id` has been changed to `status_id` in keeping with the rest of the Api and to avoid shadowing a builtin.

4.1.6 twitter.api.Api.GetBlocks()

- Method no longer accepts parameters `user_id` or `screen_name` as these are not honored by Twitter. The data returned will be for the authenticated user only.
- Parameter `cursor` is no longer accepted – this method will return **all** users being blocked by the currently authenticated user. If you need paging, please use *twitter.api.Api.GetBlocksPaged()* instead.

4.1.7 twitter.api.Api.GetFollowers()

- Method no longer honors a `count` or `cursor` parameter. These have been deprecated in favor of making this method explicitly a convenience function to return a list of every `twitter.User` who is following the specified or authenticated user. A warning will be raised if `count` or `cursor` is passed with the expectation that breaking behavior will be introduced in a later version.
- Method now takes an optional parameter of `total_count`, which limits the number of users to return. If this is not set, the data returned will be all users following the specified user.
- The kwarg `include_user_entities` now defaults to `True`. This was set to `False` previously, but would not be included in query parameters sent to Twitter. Without the query parameter in the URL, Twitter would default to returning user_entities, so this change makes this behavior explicit.

4.1.8 twitter.api.Api.GetFollowersPaged()

- The third value of the tuple returned by this method is now a list of twitter.User objects in accordance with its doc string rather than the raw data from API.
- The kwarg `include_user_entities` now defaults to `True`. This was set to `False` previously, but would not be included in query parameters sent to Twitter. Without the query parameter in the URL, Twitter would default to returning user_entities, so this change makes this behavior explicit and consistent with the previously ambiguous behavior.

4.1.9 twitter.api.Api.GetFriends()

- Method no longer honors a `count` or `cursor` parameter. These have been deprecated in favor of making this method explicitly a convenience function to return a list of every `twitter.User` who is followed by the specified or authenticated user. A warning will be raised if `count` or `cursor` is passed with the expectation that breaking behavior will be introduced in a later version.
- Method now takes an optional parameter of `total_count`, which limits the number of users to return. If this is not set, the data returned will be all users followed by the specified user.
- The kwarg `include_user_entities` now defaults to `True`. This was set to `False` previously, but would not be included in query parameters sent to Twitter. Without the query parameter in the URL, Twitter would default to returning user_entities, so this change makes this behavior explicit.

4.1.10 twitter.api.Api.GetFriendsPaged()

- The third value of the tuple returned by this method is now a list of twitter.User objects in accordance with its doc string rather than the raw data from API.
- The kwarg `include_user_entities` now defaults to `True`. This was set to `False` previously, but would not be included in query parameters sent to Twitter. Without the query parameter in the URL, Twitter would default to returning user_entities, so this change makes this behavior explicit.

4.1.11 twitter.api.Api.GetListMembers()

- No longer accepts `cursor` parameter. If you require granular control over the paging of the twitter.list.List members, please user twitter.api.Api.GetListMembersPaged instead.

4.1.12 twitter.api.Api.GetStatus()

- Kwarg `id` has been changed to `status_id` in keeping with the rest of the Api and to avoid shadowing a builtin.

4.1.13 twitter.api.Api.GetStatusOembed()

- Kwarg `id` has been changed to `status_id` in keeping with the rest of the Api and to avoid shadowing a builtin.

4.1.14 twitter.api.Api.GetSearch()

- Adds `raw_query` method. See *Raw Queries* for more information.

4.1.15 twitter.api.Api.GetTrendsWoeid()

- Kwarg `id` has been changed to `woeid` in order to avoid shadowing a builtin and be more descriptive.

4.1.16 twitter.api.Api.GetUserStream()

- Parameter 'stall_warning' is now 'stall_warnings' in line with GetStreamFilter and Twitter's naming convention. This should now actually return stall warnings, whereas it did not have any effect previously.

4.1.17 twitter.api.Api.LookupFriendship()

- Method will now accept a list for either `user_id` or `screen_name`. The list can contain either ints, strings, or `twitter.user.User` objects for either `user_id` or `screen_name`.
- Return value is a list of `twitter.user.UserStatus` objects.

4.1.18 twitter.api.Api.PostUpdate()

- Now accepts three new parameters: `media`, `media_additional_owners`, and `media_category`. `media` can be a URL, a local file, or a file-like object (something with a `read()` method), or a list of any combination of the above.
- `media_additional_owners` should be a list of user ids representing Twitter users that should be able to use the uploaded media in their tweets. If you pass a list of media, then **additional owners will apply to each object.** If you need more granular control, please use the UploadMedia* methods.
- `media_category`: Only for use with the AdsAPI. See https://dev.twitter.com/ads/creative/promoted-video-overview if this applies to your application.

4.1.19 twitter.api.Api.PostRetweet()

- Kwarg `original_id` has been changed to `status_id` in order to avoid shadowing a builtin and be more descriptive.

4.2 Deprecation

4.2.1 twitter.api.Api.PostMedia()

- This endpoint is deprecated by Twitter. Python-twitter will throw a warning about using the method and advise you to use PostUpdate() instead. There is no schedule for when this will be removed from Twitter.

4.2.2 twitter.api.Api.PostMultipleMedia()

- This method should be replaced by passing a list of media objects (either URLs, local files, or file-like objects) to PostUpdate. You are limited to a maximum of 4 media files per tweet.

4.3 New Methods

4.3.1 twitter.api.Api.GetBlocksIDs()

- Returns **all** the users currently blocked by the authenticated user as user IDs. The user IDs will be integers.

4.3.2 twitter.api.Api.GetBlocksIDsPaged()

- Returns one page, specified by the cursor parameter, of the users currently blocked by the authenticated user as user IDs.

4.3.3 twitter.api.Api.GetBlocksPaged()

- Allows you to page through the currently authenticated user's blocked users. Method returns three values: the next cursor, the previous cursor, and a list of `twitter.User` instances representing the blocked users.

4.3.4 twitter.api.Api.GetListMembersPaged()

- Allows you to page through a the members of a given twitter.list.List.
- `cursor` parameter operates as with other methods, denoting the page of members that you wish to retrieve.
- Returns `next_cursor`, `previous_cursor`, and a list containing the users that are members of the given twitter.list.List.

4.3.5 twitter.api.Api.GetListsPaged()

- Much like *twitter.api.Api.GetFriendsPaged()* and similar methods, this allows you to retrieve an arbitrary page of `twitter.list.List` for either the currently authenticated user or a user specified by `user_id` or `screen_name`.
- `cursor` should be `-1` for the first page.
- Returns the `next_cursor`, `previous_cursor`, and a list of `twitter.list.List` instances.

4.3.6 twitter.api.Api.UploadMediaChunked()

- API method allows chunked upload to upload.twitter.com. Similar to Api.PostMedia(), this method can take either a local filename (str), a URL (str), or a file-like object. The image or video type will be determined by `mimetypes` (see *twitter.twitter_utils.parse_media_file()* for details).
- Optionally, you can specify a chunk_size for uploads when instantiating the Api object. This should be given in bytes. The default is 1MB (that is, 1048576 bytes). Any chunk_size given below 16KB will result in a warning: Twitter will return an error if you try to upload more than 999 chunks of data; for example, if you are uploading a 15MB video, then a chunk_size lower than 15729 bytes will result in 1000 APPEND commands being sent to the API, so you'll get an error. 16KB seems like a reasonable lower bound, but if your use case is well-defined, then python-twitter will not enforce this behavior.
- Another thing to take into consideration: if you're working in a RAM-constrained environment, a very large chunk_size will increase your RAM usage when uploading media through this endpoint.
- The return value will be the `media_id` of the uploaded file.

4.3.7 twitter.api.Api.UploadMediaSimple()

- Provides the ability to upload a single media file to Twitter without using the ChunkedUpload endpoint. This method should be used on smaller files and reduces the roundtrips from Twitter from three (for UploadMediaChunked) to one.
- Return value is the `media_id` of the uploaded file.

CHAPTER 5

Changelog

5.1 Version 3.4.2

Bugfixes:

- Allow upload of GIFs with size up to 15mb. See #538

5.2 Version 3.4.1

Bugfixes:

- Fix an issue where *twitter.twitter_utils.calc_expected_status_length()* was failing for python 2 due to a failure to convert a bytes string to unicode. Github issue #546.
- Documentation fix for *twitter.api.Api.UsersLookup()*. UsersLookup can take a string or a list and properly parses both of them now. Github issues #535 and #549.
- Properly decode response content for *twitter.twitter_utils.http_to_file()*. Github issue #521.
- Fix an issue with loading extended_tweet entities from Streaming API where tweets would be truncated when converting to a *twitter.models.Status*. Github issues #491 and #506.

5.3 Version 3.4

5.3.1 Deprecations

- `twitter.api.Api.UpdateBackgroundImage()`. Please make sure that your code does not call this function as it will now return a hard error. There is no replacement function. This was deprecated by Twitter around July 2015.
- `twitter.api.Api.PostMedia()` has been removed. Please use *twitter.api.Api.PostUpdate()* instead.

- `twitter.api.Api.PostMultipleMedia()`. Please use *twitter.api.Api.PostUpdate()* instead.

5.4 Version 3.3.1

- Adds support for 280 character limit.

5.5 Version 3.3

- Adds application only authentication. See Twitter's documentation for details. To use application only authentication, pass *application_only_auth* when creating the Api; the bearer token will be automatically retrieved.
- Adds function `twitter.api.GetAppOnlyAuthToken()`
- Adds *filter_level* keyword argument for `twitter.api.GetStreamFilter()`, `twitter.api.GetUserStream()`
- Adds *proxies* keyword argument for creating an Api instance. Pass a dictionary of proxies for the request to pass through, if not specified allows requests lib to use environmental variables for proxy if any.
- Adds support for *quoted_status* to the *twitter.models.Status* model.

5.6 Version 3.2.1

- *twitter.twitter_utils.calc_expected_status_length()* should now function properly. Previously, URLs would be counted incorrectly. See PR #416
- *twitter.api.Api.PostUpdates()* now passes any keyword arguments on the edge case that only one tweet was actually being posted.

5.7 Version 3.2

5.7.1 Deprecations

Nothing is being deprecationed this version, however here's what's being deprecated as of v. 3.3.0:

- `twitter.api.Api.UpdateBackgroundImage()`. Please make sure that your code does not call this function as it will be returning a hard error. There is no replace function. This was deprecated by Twitter around July 2015.
- `twitter.api.Api.PostMedia()` will be removed. Please use *twitter.api.Api.PostUpdate()* instead.
- `twitter.api.Api.PostMultipleMedia()`. Please use *twitter.api.Api.PostUpdate()* instead.
- `twitter.api.GetFriends()` will no longer accept a *cursor* or *count* parameter. Please use `twitter.api.GetFriendsPaged()` instead.
- `twitter.api.GetFollowers()` will no longer accept a *cursor* or *count* parameter. Please use `twitter.api.GetFollowersPaged()` instead.

5.7.2 What's New

- We've added new deprecation warnings, so it's easier to track when things go away. All of python-twitter's deprecation warnings will be a subclass of *twitter.error.PythonTwitterDeprecationWarning* and will have a version number associated with them such as *twitter.error.PythonTwitterDeprecationWarning330*.
- *twitter.models.User* now contains a `following` attribute, which describes whether the authenticated user is following the User. PR #351
- *twitter.models.DirectMessage* contains a full *twitter.models.User* object for both the `DirectMessage.sender` and `DirectMessage.recipient` properties. PR #384.
- You can now upload Quicktime movies (`*.mov`). PR #372.
- If you have a whitelisted app, you can now get the authenticated user's email address through a call to *twitter.api.Api.VerifyCredentials()*. If your app isn't whitelisted, no error is returned. PR #376.
- Google App Engine support has been reintegrated into the library. Check out PR #383.
- *video_info* is now available on a *twitter.models.Media* object, which allows access to video urls/bitrates/etc. in the *extended_entities* node of a tweet.

5.7.3 What's Changed

- *twitter.models.Trend*'s *volume* attribute has been renamed *tweet_volume* in line with Twitter's naming convention. This change should allow users to access the number of tweets being tweeted for a given Trend. PR #375
- *twitter.ratelimit.RateLimit* should behave better now and adds a 1-second padding to requests after sleeping.
- *twitter.ratelimit.RateLimit* now keeps track of your rate limit status even if you don't have `sleep_on_rate_limit` set to `True` when instatiating the API. If you want to add different behavior on hitting a rate limit, you should be able to now by querying the rate limit object. See PR #370 for the technical details of the change. There should be no difference in behavior for the defaults, but let us know.

5.7.4 Bugfixes

- *twitter.models.Media* again contains a `sizes` attribute, which was missed back in the Version 3.0 release. PR #360
- The previously bloated *twitter.api.Api.UploadMediaChunked()* function has been broken out into three related functions and fixes two an incompatibility with python 2.7. Behavior remains the same, but this should simplify matters. PR #347
- Fix for *twitter.api.Api.PostUpdate()* where a passing an integer to the `media` parameter would cause an iteration error to occur. PR #347
- Fix for 401 errors that were occuring in the Streaming Endpoints. PR #364

5.8 Version 3.1

5.8.1 What's New

- *twitter.api.Api.PostMediaMetadata()* Method allows the posting of alt text (hover text) to a photo on Twitter. Note that it appears that you have to call this method prior to attaching the photo to a status.
- A couple new methods have been added related to showing the connections between two users:
  - *twitter.api.Api.ShowFriendship()* shows the connection between two users (i.e., are they following each other?)
  - *twitter.api.Api.IncomingFriendship()* shows all of the authenticated user's pending follower requests (if the user has set their account to private).
  - *twitter.api.Api.OutgoingFriendship()* shows the authenticated user's request to follow other users (i.e. the user has attempted to follow a private account).
- Several methods were added related to muting users:
  - *twitter.api.Api.GetMutes()* returns **all** users the currently authenticated user is muting (as `twitter.models.User` objects).
  - *twitter.api.Api.GetMutesPaged()* returns a page of `twitter.models.User` objects.
  - *twitter.api.Api.GetMutesIDs()* returns **all** of the users the currently authenticated user is muting as integers.
  - *twitter.api.Api.GetMutesIDsPaged()* returns a single page of the users the currently authenticated user is muting as integers.

5.8.2 What's Changed

- *twitter.api.Api.GetStatus()* Now accepts the keyword argument `include_ext_alt_text` which will request alt text to be included with the Status object being returned (if available). Defaults to `True`.
- `[model].__repr__()` functions have been revised for better Unicode compatibility. If you notice any weirdness, please let us know.
- *twitter.api.Api()* no longer accepts the `shortner` parameter; however, see `examples/shorten_url.py` for an example of how to use a URL shortener with the API.
- `twitter.api.Api._Encode()` and `twitter.api.Api._EncodePostData()` have both been refactored out of the API.
- *twitter.models.Media* now has an attribute `ext_alt_text` for alt (hover) text for images posted to Twitter.
- *twitter.models.Status* no longer has the properties `relative_created_at`, `now`, or `Now`. If you require a relative time, we suggest using a third-party library.
- Updated examples, specifically `examples/twitter-to-xhtml.py`, `examples/view_friends.py`, `examples/shorten_url.py`
- Updated `get_access_token.py` script to be python3 compatible.
- *twitter.api.Api.GetStreamFilter()* now accepts an optional languages parameter as a list.

CHAPTER 6

Rate Limiting

6.1 Overview

Twitter imposes rate limiting based either on user tokens or application tokens. Please see: API Rate Limits for a more detailed explanation of Twitter's policies. What follows will be a summary of how Python-Twitter attempts to deal with rate limits and how you should expect those limits to be respected (or not).

Python-Twitter tries to abstract away the details of Twitter's rate limiting by allowing you to globally respect those limits or ignore them. If you wish to have the application sleep when it hits a rate limit, you should instantiate the API with `sleep_on_rate_limit=True` like so:

```
import twitter
api = twitter.Api(consumer_key=[consumer key],
                  consumer_secret=[consumer secret],
                  access_token_key=[access token],
                  access_token_secret=[access token secret],
                  sleep_on_rate_limit=True)
```

By default, python-twitter will raise a hard error for rate limits

Effectively, when the API determines that the **next** call to an endpoint will result in a rate limit error being thrown by Twitter, it will sleep until you are able to safely make that call. For most API methods, the headers in the response from Twitter will contain the following information:

`x-rate-limit-limit`: The number of times you can request the given endpoint within a certain number of minutes (otherwise known as a window).

`x-rate-limit-remaining`: The number of times you have left for a given endpoint within a window.

`x-rate-limit-reset`: The number of seconds left until the window resets.

For most endpoints, this is 15 requests per 15 minutes. So if you have set the global `sleep_on_rate_limit` to `True`, the process looks something like this:

```
api.GetListMembersPaged()
# GET /list/{id}/members.json?cursor=-1
# GET /list/{id}/members.json?cursor=2
# GET /list/{id}/members.json?cursor=3
# GET /list/{id}/members.json?cursor=4
# GET /list/{id}/members.json?cursor=5
# GET /list/{id}/members.json?cursor=6
# GET /list/{id}/members.json?cursor=7
# GET /list/{id}/members.json?cursor=8
# GET /list/{id}/members.json?cursor=9
# GET /list/{id}/members.json?cursor=10
# GET /list/{id}/members.json?cursor=11
# GET /list/{id}/members.json?cursor=12
# GET /list/{id}/members.json?cursor=13
# GET /list/{id}/members.json?cursor=14

# This last GET request returns a response where x-rate-limit-remaining
# is equal to 0, so the API sleeps for 15 minutes

# GET /list/{id}/members.json?cursor=15

# ... etc ...
```

If you would rather not have your API instance sleep when hitting, then do not pass `sleep_on_rate_limit=True` to your API instance. This will cause the API to raise a hard error when attempting to make call #15 above.

6.2 Technical

The `twitter/ratelimit.py` file contains the code that handles storing and checking rate limits for endpoints. Since Twitter does not send any information regarding the endpoint that you are requesting with the `x-rate-limit-*` headers, the endpoint is determined by some regex using the URL.

The twitter.Api instance contains an `Api.rate_limit` object that you can inspect to see the current limits for any URL and exposes a number of methods for querying and setting rate limits on a per-resource (i.e., endpoint) basis. See *twitter.ratelimit.RateLimit()* for more information.

CHAPTER 7

Models

Python-twitter provides the following models of the objects returned by the Twitter API:

- *twitter.models.Category*
- *twitter.models.DirectMessage*
- *twitter.models.Hashtag*
- *twitter.models.List*
- *twitter.models.Media*
- *twitter.models.Status*
- *twitter.models.Trend*
- *twitter.models.Url*
- *twitter.models.User*
- *twitter.models.UserStatus*

CHAPTER 8

Searching

8.1 Raw Queries

To the `Api.GetSearch()` method, you can pass the parameter `raw_query`, which should be the query string you wish to use for the search **omitting the leading "?"**. This will override every other parameter. Twitter's search parameters are quite complex, so if you have a need for a very particular search, you can find Twitter's documentation at https://dev.twitter.com/rest/public/search.

For example, if you want to search for only tweets containing the word "twitter", then you could do the following:

```
results = api.GetSearch(
    raw_query="q=twitter%20&result_type=recent&since=2014-07-19&count=100")
```

If you want to build a search query and you're not quite sure how it should look all put together, you can use Twitter's Advanced Search tool: https://twitter.com/search-advanced, and then use the part of search URL after the "?" to use for the Api, removing the `&src=typd` portion.

CHAPTER 9

Using with Django

Additional template tags that expand tweet urls and urlize tweet text. See the django template tags available for use with python-twitter: https://github.com/radzhome/python-twitter-django-tags

CHAPTER 10

Modules Documentation

10.1 API

A library that provides a Python interface to the Twitter API

class twitter.api.**Api**(*consumer_key=None*, *consumer_secret=None*, *access_token_key=None*, *access_token_secret=None*, *application_only_auth=False*, *input_encoding=None*, *request_headers=None*, *cache=<object object>*, *base_url=None*, *stream_url=None*, *upload_url=None*, *chunk_size=1048576*, *use_gzip_compression=False*, *debugHTTP=False*, *timeout=None*, *sleep_on_rate_limit=False*, *tweet_mode='compat'*, *proxies=None*)

Bases: object

A python interface into the Twitter API

By default, the Api caches results for 1 minute.

Example usage:

To create an instance of the twitter.Api class, with no authentication:

```
>>> import twitter
>>> api = twitter.Api()
```

To fetch a single user's public status messages, where "user" is either a Twitter "short name" or their user id.

```
>>> statuses = api.GetUserTimeline(user)
>>> print([s.text for s in statuses])
```

To use authentication, instantiate the twitter.Api class with a consumer key and secret; and the oAuth key and secret:

```
>>> api = twitter.Api(consumer_key='twitter consumer key',
                      consumer_secret='twitter consumer secret',
```

(continues on next page)

(continued from previous page)

```
                        access_token_key='the_key_given',
                        access_token_secret='the_key_secret')
```

To fetch your friends (after being authenticated):

```
>>> users = api.GetFriends()
>>> print([u.name for u in users])
```

To post a twitter status message (after being authenticated):

```
>>> status = api.PostUpdate('I love python-twitter!')
>>> print(status.text)
I love python-twitter!
```

There are many other methods, including:

```
>>> api.PostUpdates(status)
>>> api.PostDirectMessage(user, text)
>>> api.GetUser(user)
>>> api.GetReplies()
>>> api.GetUserTimeline(user)
>>> api.GetHomeTimeline()
>>> api.GetStatus(status_id)
>>> api.GetStatuses(status_ids)
>>> api.DestroyStatus(status_id)
>>> api.GetFriends(user)
>>> api.GetFollowers()
>>> api.GetFeatured()
>>> api.GetDirectMessages()
>>> api.GetSentDirectMessages()
>>> api.PostDirectMessage(user, text)
>>> api.DestroyDirectMessage(message_id)
>>> api.DestroyFriendship(user)
>>> api.CreateFriendship(user)
>>> api.LookupFriendship(user)
>>> api.VerifyCredentials()
```

CheckRateLimit(*url*)
Checks a URL to see the rate limit status for that endpoint.

Parameters **url** (*str*) – URL to check against the current rate limits.

Returns EndpointRateLimit namedtuple.

Return type namedtuple

ClearCredentials()
Clear any credentials for this instance

CreateBlock(*user_id=None*, *screen_name=None*, *include_entities=True*, *skip_status=False*)
Blocks the user specified by either user_id or screen_name.

Parameters

- **user_id** (*int, optional*) – The numerical ID of the user to block.
- **screen_name** (*str, optional*) – The screen name of the user to block.
- **include_entities** (*bool, optional*) – The entities node will not be included if set to False.

- **skip_status** (*bool, optional*) – When set to False, the blocked User's statuses will not be included with the returned User object.

Returns A twitter.User instance representing the blocked user.

CreateFavorite(*status=None*, *status_id=None*, *include_entities=True*)
Favorites the specified status object or id as the authenticating user.

Returns the favorite status when successful.

Parameters

- **status_id** (*int, optional*) – The id of the twitter status to mark as a favorite.
- **status** (*twitter.Status, optional*) – The twitter.Status object to mark as a favorite.
- **include_entities** (*bool, optional*) – The entities node will be omitted when set to False.

Returns A twitter.Status instance representing the newly-marked favorite.

CreateFriendship(*user_id=None*, *screen_name=None*, *follow=True*, *retweets=True*, ***kwargs*)
Befriends the user specified by the user_id or screen_name.

Parameters

- **user_id** (*int, optional*) – A user_id to follow
- **screen_name** (*str, optional*) – A screen_name to follow
- **follow** (*bool, optional*) – Set to False to disable notifications for the target user
- **retweets** (*bool, optional*) – Enable or disable retweets from the target user.

Returns A twitter.User instance representing the befriended user.

CreateList(*name*, *mode=None*, *description=None*)
Creates a new list with the give name for the authenticated user.

Parameters

- **name** (*str*) – New name for the list
- **mode** (*str, optional*) – 'public' or 'private'. Defaults to 'public'.
- **description** (*str, optional*) – Description of the list.

Returns A twitter.List instance representing the new list

Return type twitter.list.List

CreateListsMember(*list_id=None*, *slug=None*, *user_id=None*, *screen_name=None*, *owner_screen_name=None*, *owner_id=None*)
Add a new member (or list of members) to the specified list.

Parameters

- **list_id** (*int, optional*) – The numerical id of the list.
- **slug** (*str, optional*) – You can identify a list by its slug instead of its numerical id. If you decide to do so, note that you'll also have to specify the list owner using the owner_id or owner_screen_name parameters.
- **user_id** (*int, optional*) – The user_id or a list of user_id's to add to the list. If not given, then screen_name is required.

- **screen_name** (*str, optional*) – The screen_name or a list of screen_name's to add to the list. If not given, then user_id is required.
- **owner_screen_name** (*str, optional*) – The screen_name of the user who owns the list being requested by a slug.
- **owner_id** (*int, optional*) – The user ID of the user who owns the list being requested by a slug.

Returns A twitter.List instance representing the list subscribed to.

Return type twitter.list.List

CreateMute(*user_id=None*, *screen_name=None*, *include_entities=True*, *skip_status=False*)
Mutes the user specified by either user_id or screen_name.

Parameters

- **user_id** (*int, optional*) – The numerical ID of the user to mute.
- **screen_name** (*str, optional*) – The screen name of the user to mute.
- **include_entities** (*bool, optional*) – The entities node will not be included if set to False.
- **skip_status** (*bool, optional*) – When set to False, the muted User's statuses will not be included with the returned User object.

Returns A twitter.User instance representing the muted user.

CreateSubscription(*owner_screen_name=None*, *owner_id=None*, *list_id=None*, *slug=None*)
Creates a subscription to a list by the authenticated user.

Parameters

- **owner_screen_name** (*str, optional*) – The screen_name of the user who owns the list being requested by a slug.
- **owner_id** (*int, optional*) – The user ID of the user who owns the list being requested by a slug.
- **list_id** (*int, optional*) – The numerical id of the list.
- **slug** (*str, optional*) – You can identify a list by its slug instead of its numerical id. If you decide to do so, note that you'll also have to specify the list owner using the owner_id or owner_screen_name parameters.

Returns A twitter.User instance representing the user subscribed

Return type twitter.user.User

DEFAULT_CACHE_TIMEOUT = 60

DestroyBlock(*user_id=None*, *screen_name=None*, *include_entities=True*, *skip_status=False*)
Unlocks the user specified by either user_id or screen_name.

Parameters

- **user_id** (*int, optional*) – The numerical ID of the user to block.
- **screen_name** (*str, optional*) – The screen name of the user to block.
- **include_entities** (*bool, optional*) – The entities node will not be included if set to False.
- **skip_status** (*bool, optional*) – When set to False, the blocked User's statuses will not be included with the returned User object.

Returns A twitter.User instance representing the blocked user.

DestroyDirectMessage(*message_id*, *include_entities=True*, *return_json=False*)
Destroys the direct message specified in the required ID parameter.

The twitter.Api instance must be authenticated, and the authenticating user must be the recipient of the specified direct message.

Parameters

- **message_id** – The id of the direct message to be destroyed
- **return_json** (*bool, optional*) – If True JSON data will be returned, instead of twitter.User

Returns A twitter.DirectMessage instance representing the message destroyed

DestroyFavorite(*status=None*, *status_id=None*, *include_entities=True*)
Un-Favorites the specified status object or id as the authenticating user.

Returns the un-favorited status when successful.

Parameters

- **status_id** (*int, optional*) – The id of the twitter status to mark as a favorite.
- **status** (*twitter.Status, optional*) – The twitter.Status object to mark as a favorite.
- **include_entities** (*bool, optional*) – The entities node will be omitted when set to False.

Returns A twitter.Status instance representing the newly-unmarked favorite.

DestroyFriendship(*user_id=None*, *screen_name=None*)
Discontinues friendship with a user_id or screen_name.

Parameters

- **user_id** – A user_id to unfollow [Optional]
- **screen_name** – A screen_name to unfollow [Optional]

Returns A twitter.User instance representing the discontinued friend.

DestroyList(*owner_screen_name=None*, *owner_id=None*, *list_id=None*, *slug=None*)
Destroys the list identified by list_id or slug and one of owner_screen_name or owner_id.

Parameters

- **owner_screen_name** (*str, optional*) – The screen_name of the user who owns the list being requested by a slug.
- **owner_id** (*int, optional*) – The user ID of the user who owns the list being requested by a slug.
- **list_id** (*int, optional*) – The numerical id of the list.
- **slug** (*str, optional*) – You can identify a list by its slug instead of its numerical id. If you decide to do so, note that you'll also have to specify the list owner using the owner_id or owner_screen_name parameters.

Returns A twitter.List instance representing the removed list.

Return type twitter.list.List

DestroyListsMember(*list_id=None*, *slug=None*, *owner_screen_name=None*, *owner_id=None*, *user_id=None*, *screen_name=None*)

Destroys the subscription to a list for the authenticated user.

Parameters

- **list_id** (*int, optional*) – The numerical id of the list.
- **slug** (*str, optional*) – You can identify a list by its slug instead of its numerical id. If you decide to do so, note that you'll also have to specify the list owner using the owner_id or owner_screen_name parameters.
- **owner_screen_name** (*str, optional*) – The screen_name of the user who owns the list being requested by a slug.
- **owner_id** (*int, optional*) – The user ID of the user who owns the list being requested by a slug.
- **user_id** (*int, optional*) – The user_id or a list of user_id's to remove from the list. If not given, then screen_name is required.
- **screen_name** (*str, optional*) – The screen_name or a list of Screen_name's to remove from the list. If not given, then user_id is required.

Returns A twitter.List instance representing the removed list.

Return type twitter.list.List

DestroyMute(*user_id=None*, *screen_name=None*, *include_entities=True*, *skip_status=False*)

Unlocks the user specified by either user_id or screen_name.

Parameters

- **user_id** (*int, optional*) – The numerical ID of the user to mute.
- **screen_name** (*str, optional*) – The screen name of the user to mute.
- **include_entities** (*bool, optional*) – The entities node will not be included if set to False.
- **skip_status** (*bool, optional*) – When set to False, the muted User's statuses will not be included with the returned User object.

Returns A twitter.User instance representing the muted user.

DestroyStatus(*status_id*, *trim_user=False*)

Destroys the status specified by the required ID parameter.

The authenticating user must be the author of the specified status.

Parameters

- **status_id** (*int*) – The numerical ID of the status you're trying to destroy.
- **trim_user** (*bool, optional*) – When set to True, each tweet returned in a timeline will include a user object including only the status authors numerical ID.

Returns A twitter.Status instance representing the destroyed status message

DestroySubscription(*owner_screen_name=None*, *owner_id=None*, *list_id=None*, *slug=None*)

Destroys the subscription to a list for the authenticated user.

Parameters

- **owner_screen_name** (*str, optional*) – The screen_name of the user who owns the list being requested by a slug.

- **owner_id** (*int, optional*) – The user ID of the user who owns the list being requested by a slug.
- **list_id** (*int, optional*) – The numerical id of the list.
- **slug** (*str, optional*) – You can identify a list by its slug instead of its numerical id. If you decide to do so, note that you'll also have to specify the list owner using the owner_id or owner_screen_name parameters.

Returns A twitter.List instance representing the removed list.

Return type twitter.list.List

static GetAppOnlyAuthToken(*consumer_key*, *consumer_secret*)
Generate a Bearer Token from consumer_key and consumer_secret

GetBlocks(*skip_status=False*, *include_entities=False*)
Fetch the sequence of all users (as twitter.User instances), blocked by the currently authenticated user.

Parameters

- **skip_status** (*bool, optional*) – If True the statuses will not be returned in the user items.
- **include_entities** (*bool, optional*) – When True, the user entities will be included.

Returns A list of twitter.User instances, one for each blocked user.

GetBlocksIDs(*stringify_ids=False*)
Fetch the sequence of all user IDs blocked by the currently authenticated user.

Parameters **stringify_ids** (*bool, optional*) – If True user IDs will be returned as strings rather than integers.

Returns A list of user IDs for all blocked users.

GetBlocksIDsPaged(*cursor=-1*, *stringify_ids=False*)
Fetch a page of the user IDs blocked by the currently authenticated user.

Parameters

- **cursor** (*int, optional*) – Should be set to -1 if you want the first page, thereafter denotes the page of blocked users that you want to return.
- **stringify_ids** (*bool, optional*) – If True user IDs will be returned as strings rather than integers.

Returns next_cursor, previous_cursor, list of user IDs of blocked users.

GetBlocksPaged(*cursor=-1*, *skip_status=False*, *include_entities=False*)
Fetch a page of the users (as twitter.User instances) blocked by the currently authenticated user.

Parameters

- **cursor** (*int, optional*) – Should be set to -1 if you want the first page, thereafter denotes the page of blocked users that you want to return.
- **skip_status** (*bool, optional*) – If True the statuses will not be returned in the user items.
- **include_entities** (*bool, optional*) – When True, the user entities will be included.

Returns next_cursor, previous_cursor, list of twitter.User instances, one for each blocked user.

GetDirectMessages(*since_id=None*, *max_id=None*, *count=None*, *include_entities=True*, *skip_status=False*, *full_text=False*, *page=None*, *return_json=False*)

Returns a list of the direct messages sent to the authenticating user.

Parameters

- **since_id** – Returns results with an ID greater than (that is, more recent than) the specified ID. There are limits to the number of Tweets which can be accessed through the API. If the limit of Tweets has occurred since the since_id, the since_id will be forced to the oldest ID available. [Optional]
- **max_id** – Returns results with an ID less than (that is, older than) or equal to the specified ID. [Optional]
- **count** – Specifies the number of direct messages to try and retrieve, up to a maximum of 200. The value of count is best thought of as a limit to the number of Tweets to return because suspended or deleted content is removed after the count has been applied. [Optional]
- **include_entities** – The entities node will be omitted when set to False. [Optional]
- **skip_status** – When set to True statuses will not be included in the returned user objects. [Optional]
- **full_text** – When set to True full message will be included in the returned message object if message length is bigger than CHARACTER_LIMIT characters. [Optional]
- **page** – If you want more than 200 messages, you can use this and get 20 messages each time. You must recall it and increment the page value until it return nothing. You can't use count option with it. First value is 1 and not 0.
- **return_json** (*bool, optional*) – If True JSON data will be returned, instead of twitter.User

Returns A sequence of twitter.DirectMessage instances

GetFavorites(*user_id=None*, *screen_name=None*, *count=None*, *since_id=None*, *max_id=None*, *include_entities=True*, *return_json=False*)

Return a list of Status objects representing favorited tweets.

Returns up to 200 most recent tweets for the authenticated user.

Parameters

- **user_id** (*int, optional*) – Specifies the ID of the user for whom to return the favorites. Helpful for disambiguating when a valid user ID is also a valid screen name.
- **screen_name** (*str, optional*) – Specifies the screen name of the user for whom to return the favorites. Helpful for disambiguating when a valid screen name is also a user ID.
- **since_id** (*int, optional*) – Returns results with an ID greater than (that is, more recent than) the specified ID. There are limits to the number of Tweets which can be accessed through the API. If the limit of Tweets has occurred since the since_id, the since_id will be forced to the oldest ID available.
- **max_id** (*int, optional*) – Returns only statuses with an ID less than (that is, older than) or equal to the specified ID.
- **count** (*int, optional*) – Specifies the number of statuses to retrieve. May not be greater than 200.
- **include_entities** (*bool, optional*) – The entities node will be omitted when set to False.

- **return_json** (*bool, optional*) – If True JSON data will be returned, instead of twitter.User

Returns A sequence of Status instances, one for each favorited tweet up to count

GetFollowerIDs(*user_id=None*, *screen_name=None*, *cursor=None*, *stringify_ids=False*, *count=None*, *total_count=None*)

Returns a list of twitter user id's for every person that is following the specified user.

Parameters

- **user_id** – The id of the user to retrieve the id list for. [Optional]
- **screen_name** – The screen_name of the user to retrieve the id list for. [Optional]
- **cursor** – Specifies the Twitter API Cursor location to start at. Note: there are pagination limits. [Optional]
- **stringify_ids** – if True then twitter will return the ids as strings instead of integers. [Optional]
- **count** – The number of user id's to retrieve per API request. Please be aware that this might get you rate-limited if set to a small number. By default Twitter will retrieve 5000 UIDs per call. [Optional]
- **total_count** – The total amount of UIDs to retrieve. Good if the account has many followers and you don't want to get rate limited. The data returned might contain more UIDs if total_count is not a multiple of count (5000 by default). [Optional]

Returns A list of integers, one for each user id.

GetFollowerIDsPaged(*user_id=None*, *screen_name=None*, *cursor=-1*, *stringify_ids=False*, *count=5000*)

Make a cursor driven call to return a list of one page followers.

The caller is responsible for handling the cursor value and looping to gather all of the data

Parameters

- **user_id** – The twitter id of the user whose followers you are fetching. If not specified, defaults to the authenticated user. [Optional]
- **screen_name** – The twitter name of the user whose followers you are fetching. If not specified, defaults to the authenticated user. [Optional]
- **cursor** – Should be set to -1 for the initial call and then is used to control what result page Twitter returns.
- **stringify_ids** – if True then twitter will return the ids as strings instead of integers. [Optional]
- **count** – The number of user id's to retrieve per API request. Please be aware that this might get you rate-limited if set to a small number. By default Twitter will retrieve 5000 UIDs per call. [Optional]

Returns next_cursor, previous_cursor, data sequence of user ids, one for each follower

GetFollowers(*user_id=None*, *screen_name=None*, *cursor=None*, *count=None*, *total_count=None*, *skip_status=False*, *include_user_entities=True*)

Fetch the sequence of twitter.User instances, one for each follower.

If both user_id and screen_name are specified, this call will return the followers of the user specified by screen_name, however this behavior is undocumented by Twitter and may change without warning.

Parameters

- **user_id** – The twitter id of the user whose followers you are fetching. If not specified, defaults to the authenticated user. [Optional]
- **screen_name** – The twitter name of the user whose followers you are fetching. If not specified, defaults to the authenticated user. [Optional]
- **cursor** – Should be set to -1 for the initial call and then is used to control what result page Twitter returns.
- **count** – The number of users to return per page, up to a maximum of 200. Defaults to 200. [Optional]
- **total_count** – The upper bound of number of users to return, defaults to None.
- **skip_status** – If True the statuses will not be returned in the user items. [Optional]
- **include_user_entities** – When True, the user entities will be included. [Optional]

Returns A sequence of twitter.User instances, one for each follower

GetFollowersPaged(*user_id=None*, *screen_name=None*, *cursor=-1*, *count=200*, *skip_status=False*, *include_user_entities=True*)

Make a cursor driven call to return the list of all followers

Parameters

- **user_id** – The twitter id of the user whose followers you are fetching. If not specified, defaults to the authenticated user. [Optional]
- **screen_name** – The twitter name of the user whose followers you are fetching. If not specified, defaults to the authenticated user. [Optional]
- **cursor** – Should be set to -1 for the initial call and then is used to control what result page Twitter returns.
- **count** – The number of users to return per page, up to a maximum of 200. Defaults to 200. [Optional]
- **skip_status** – If True the statuses will not be returned in the user items. [Optional]
- **include_user_entities** – When True, the user entities will be included. [Optional]

Returns next_cursor, previous_cursor, data sequence of twitter.User instances, one for each follower

GetFriendIDs(*user_id=None*, *screen_name=None*, *cursor=None*, *count=None*, *stringify_ids=False*, *total_count=None*)

Fetch a sequence of user ids, one for each friend. Returns a list of all the given user's friends' IDs. If no user_id or screen_name is given, the friends will be those of the authenticated user.

Parameters

- **user_id** – The id of the user to retrieve the id list for. [Optional]
- **screen_name** – The screen_name of the user to retrieve the id list for. [Optional]
- **cursor** – Specifies the Twitter API Cursor location to start at. Note: there are pagination limits. [Optional]
- **stringify_ids** – if True then twitter will return the ids as strings instead of integers. [Optional]
- **count** – The number of user id's to retrieve per API request. Please be aware that this might get you rate-limited if set to a small number. By default Twitter will retrieve 5000 UIDs per call. [Optional]

- **total_count** – The total amount of UIDs to retrieve. Good if the account has many followers and you don't want to get rate limited. The data returned might contain more UIDs if total_count is not a multiple of count (5000 by default). [Optional]

Returns A list of integers, one for each user id.

GetFriendIDsPaged(*user_id=None*, *screen_name=None*, *cursor=-1*, *stringify_ids=False*, *count=5000*)

Make a cursor driven call to return the list of all friends

The caller is responsible for handling the cursor value and looping to gather all of the data

Parameters

- **user_id** – The twitter id of the user whose friends you are fetching. If not specified, defaults to the authenticated user. [Optional]
- **screen_name** – The twitter name of the user whose friends you are fetching. If not specified, defaults to the authenticated user. [Optional]
- **cursor** – Should be set to -1 for the initial call and then is used to control what result page Twitter returns.
- **stringify_ids** – if True then twitter will return the ids as strings instead of integers. [Optional]
- **count** – The number of user id's to retrieve per API request. Please be aware that this might get you rate-limited if set to a small number. By default Twitter will retrieve 5000 UIDs per call. [Optional]

Returns next_cursor, previous_cursor, data sequence of twitter.User instances, one for each friend

GetFriends(*user_id=None*, *screen_name=None*, *cursor=None*, *count=None*, *total_count=None*, *skip_status=False*, *include_user_entities=True*)

Fetch the sequence of twitter.User instances, one for each friend.

If both user_id and screen_name are specified, this call will return the followers of the user specified by screen_name, however this behavior is undocumented by Twitter and may change without warning.

Parameters

- **user_id** – The twitter id of the user whose friends you are fetching. If not specified, defaults to the authenticated user. [Optional]
- **screen_name** – The twitter name of the user whose friends you are fetching. If not specified, defaults to the authenticated user. [Optional]
- **cursor** – Should be set to -1 for the initial call and then is used to control what result page Twitter returns.
- **count** – The number of users to return per page, up to a maximum of 200. Defaults to 200. [Optional]
- **total_count** – The upper bound of number of users to return, defaults to None.
- **skip_status** – If True the statuses will not be returned in the user items. [Optional]
- **include_user_entities** – When True, the user entities will be included. [Optional]

Returns A sequence of twitter.User instances, one for each friend

GetFriendsPaged(*user_id=None*, *screen_name=None*, *cursor=-1*, *count=200*, *skip_status=False*, *include_user_entities=True*)

Make a cursor driven call to return the list of all friends.

Parameters

- **user_id** – The twitter id of the user whose friends you are fetching. If not specified, defaults to the authenticated user. [Optional]
- **screen_name** – The twitter name of the user whose friends you are fetching. If not specified, defaults to the authenticated user. [Optional]
- **cursor** – Should be set to -1 for the initial call and then is used to control what result page Twitter returns.
- **count** – The number of users to return per page, up to a current maximum of 200. Defaults to 200. [Optional]
- **skip_status** – If True the statuses will not be returned in the user items. [Optional]
- **include_user_entities** – When True, the user entities will be included. [Optional]

Returns next_cursor, previous_cursor, data sequence of twitter.User instances, one for each follower

GetHelpConfiguration()

Get basic help configuration details from Twitter.

Parameters **None** –

Returns Sets self._config and returns dict of help config values.

Return type dict

GetHomeTimeline(*count=None*, *since_id=None*, *max_id=None*, *trim_user=False*, *exclude_replies=False*, *contributor_details=False*, *include_entities=True*)

Fetch a collection of the most recent Tweets and retweets posted by the authenticating user and the users they follow.

The home timeline is central to how most users interact with Twitter.

Parameters

- **count** – Specifies the number of statuses to retrieve. May not be greater than 200. Defaults to 20. [Optional]
- **since_id** – Returns results with an ID greater than (that is, more recent than) the specified ID. There are limits to the number of Tweets which can be accessed through the API. If the limit of Tweets has occurred since the since_id, the since_id will be forced to the oldest ID available. [Optional]
- **max_id** – Returns results with an ID less than (that is, older than) or equal to the specified ID. [Optional]
- **trim_user** – When True, each tweet returned in a timeline will include a user object including only the status authors numerical ID. Omit this parameter to receive the complete user object. [Optional]
- **exclude_replies** – This parameter will prevent replies from appearing in the returned timeline. Using exclude_replies with the count parameter will mean you will receive up-to count tweets - this is because the count parameter retrieves that many tweets before filtering out retweets and replies. [Optional]
- **contributor_details** – This parameter enhances the contributors element of the status response to include the screen_name of the contributor. By default only the user_id of the contributor is included. [Optional]

- **include_entities** – The entities node will be disincluded when set to false. This node offers a variety of metadata about the tweet in a discreet structure, including: user_mentions, urls, and hashtags. [Optional]

Returns A sequence of twitter.Status instances, one for each message

GetListMembers(*list_id=None*, *slug=None*, *owner_id=None*, *owner_screen_name=None*, *skip_status=False*, *include_entities=False*)

Fetch the sequence of twitter.User instances, one for each member of the given list_id or slug.

Parameters

- **list_id** (*int, optional*) – Specifies the ID of the list to retrieve.
- **slug** (*str, optional*) – The slug name for the list to retrieve. If you specify None for the list_id, then you have to provide either a owner_screen_name or owner_id.
- **owner_id** (*int, optional*) – Specifies the ID of the user for whom to return the list timeline. Helpful for disambiguating when a valid user ID is also a valid screen name.
- **owner_screen_name** (*str, optional*) – Specifies the screen name of the user for whom to return the user_timeline. Helpful for disambiguating when a valid screen name is also a user ID.
- **skip_status** (*bool, optional*) – If True the statuses will not be returned in the user items.
- **include_entities** (*bool, optional*) – If False, the timeline will not contain additional metadata. Defaults to True.

Returns A sequence of twitter.user.User instances, one for each member of the twitter.list.List.

Return type list

GetListMembersPaged(*list_id=None*, *slug=None*, *owner_id=None*, *owner_screen_name=None*, *cursor=-1*, *count=100*, *skip_status=False*, *include_entities=True*)

Fetch the sequence of twitter.User instances, one for each member of the given list_id or slug.

Parameters

- **list_id** (*int, optional*) – Specifies the ID of the list to retrieve.
- **slug** (*str, optional*) – The slug name for the list to retrieve. If you specify None for the list_id, then you have to provide either a owner_screen_name or owner_id.
- **owner_id** (*int, optional*) – Specifies the ID of the user for whom to return the list timeline. Helpful for disambiguating when a valid user ID is also a valid screen name.
- **owner_screen_name** (*str, optional*) – Specifies the screen name of the user for whom to return the user_timeline. Helpful for disambiguating when a valid screen name is also a user ID.
- **cursor** (*int, optional*) – Should be set to -1 for the initial call and then is used to control what result page Twitter returns.
- **skip_status** (*bool, optional*) – If True the statuses will not be returned in the user items.
- **include_entities** (*bool, optional*) – If False, the timeline will not contain additional metadata. Defaults to True.

Returns A sequence of twitter.user.User instances, one for each member of the twitter.list.List.

Return type list

GetListTimeline(*list_id=None*, *slug=None*, *owner_id=None*, *owner_screen_name=None*, *since_id=None*, *max_id=None*, *count=None*, *include_rts=True*, *include_entities=True*, *return_json=False*)

Fetch the sequence of Status messages for a given List ID.

Parameters

- **list_id** (*int, optional*) – Specifies the ID of the list to retrieve.
- **slug** (*str, optional*) – The slug name for the list to retrieve. If you specify None for the list_id, then you have to provide either a owner_screen_name or owner_id.
- **owner_id** (*int, optional*) – Specifies the ID of the user for whom to return the list timeline. Helpful for disambiguating when a valid user ID is also a valid screen name.
- **owner_screen_name** (*str, optional*) – Specifies the screen name of the user for whom to return the user_timeline. Helpful for disambiguating when a valid screen name is also a user ID.
- **since_id** (*int, optional*) – Returns results with an ID greater than (that is, more recent than) the specified ID. There are limits to the number of Tweets which can be accessed through the API. If the limit of Tweets has occurred since the since_id, the since_id will be forced to the oldest ID available.
- **max_id** (*int, optional*) – Returns only statuses with an ID less than (that is, older than) or equal to the specified ID.
- **count** (*int, optional*) – Specifies the number of statuses to retrieve. May not be greater than 200.
- **include_rts** (*bool, optional*) – If True, the timeline will contain native retweets (if they exist) in addition to the standard stream of tweets.
- **include_entities** (*bool, optional*) – If False, the timeline will not contain additional metadata. Defaults to True.
- **return_json** (*bool, optional*) – If True JSON data will be returned, instead of twitter.User

Returns A list of twitter.status.Status instances, one for each message up to count.

Return type list

GetLists(*user_id=None*, *screen_name=None*)

Fetch the sequence of lists for a user. If no user_id or screen_name is passed, the data returned will be for the authenticated user.

Parameters

- **user_id** – The ID of the user for whom to return results for. [Optional]
- **screen_name** – The screen name of the user for whom to return results for. [Optional]
- **count** – The amount of results to return per page. No more than 1000 results will ever be returned in a single page. Defaults to 20. [Optional]
- **cursor** – The "page" value that Twitter will use to start building the list sequence from. Use the value of -1 to start at the beginning. Twitter will return in the result the values for next_cursor and previous_cursor. [Optional]

Returns A sequence of twitter.List instances, one for each list

GetListsList(*screen_name=None*, *user_id=None*, *reverse=False*, *return_json=False*)

Returns all lists the user subscribes to, including their own. If no user_id or screen_name is specified, the data returned will be for the authenticated user.

Parameters

- **screen_name** (*str, optional*) – Specifies the screen name of the user for whom to return the user_timeline. Helpful for disambiguating when a valid screen name is also a user ID.
- **user_id** (*int, optional*) – Specifies the ID of the user for whom to return the user_timeline. Helpful for disambiguating when a valid user ID is also a valid screen name.
- **reverse** (*bool, optional*) – If False, the owned lists will be returned first, othewise subscribed lists will be at the top. Returns a maximum of 100 entries regardless. Defaults to False.
- **return_json** (*bool, optional*) – If True JSON data will be returned, instead of twitter.User

Returns A sequence of twitter.List instances.

Return type list

GetListsPaged(*user_id=None*, *screen_name=None*, *cursor=-1*, *count=20*)

Fetch the sequence of lists for a user. If no user_id or screen_name is passed, the data returned will be for the authenticated user.

Parameters

- **user_id** (*int, optional*) – The ID of the user for whom to return results for.
- **screen_name** (*str, optional*) – The screen name of the user for whom to return results for.
- **count** (*int, optional*) – The amount of results to return per page. No more than 1000 results will ever be returned in a single page. Defaults to 20.
- **cursor** (*int, optional*) – The "page" value that Twitter will use to start building the list sequence from. Use the value of -1 to start at the beginning. Twitter will return in the result the values for next_cursor and previous_cursor.

Returns next_cursor (int), previous_cursor (int), list of twitter.List instances, one for each list

GetMemberships(*user_id=None*, *screen_name=None*, *count=20*, *cursor=-1*, *filter_to_owned_lists=False*, *return_json=False*)

Obtain the lists the specified user is a member of. If no user_id or screen_name is specified, the data returned will be for the authenticated user.

Returns a maximum of 20 lists per page by default.

Parameters

- **user_id** (*int, optional*) – The ID of the user for whom to return results for.
- **screen_name** (*str, optional*) – The screen name of the user for whom to return results for.
- **count** (*int, optional*) – The amount of results to return per page. No more than 1000 results will ever be returned in a single page. Defaults to 20.
- **cursor** (*int, optional*) – The "page" value that Twitter will use to start building the list sequence from. Use the value of -1 to start at the beginning. Twitter will return in the result the values for next_cursor and previous_cursor.
- **filter_to_owned_lists** (*bool, optional*) – Set to True to return only the lists the authenticating user owns, and the user specified by user_id or screen_name is a member of. Default value is False.

- **return_json** (*bool, optional*) – If True JSON data will be returned, instead of twitter.User

Returns A list of twitter.List instances, one for each list in which the user specified by user_id or screen_name is a member

Return type list

GetMentions(*count=None*, *since_id=None*, *max_id=None*, *trim_user=False*, *contributor_details=False*, *include_entities=True*, *return_json=False*)

Returns the 20 most recent mentions (status containing @screen_name) for the authenticating user.

Parameters

- **count** – Specifies the number of tweets to try and retrieve, up to a maximum of 200. The value of count is best thought of as a limit to the number of tweets to return because suspended or deleted content is removed after the count has been applied. [Optional]
- **since_id** – Returns results with an ID greater than (that is, more recent than) the specified ID. There are limits to the number of Tweets which can be accessed through the API. If the limit of Tweets has occurred since the since_id, the since_id will be forced to the oldest ID available. [Optional]
- **max_id** – Returns only statuses with an ID less than (that is, older than) the specified ID. [Optional]
- **trim_user** – When set to True, each tweet returned in a timeline will include a user object including only the status authors numerical ID. Omit this parameter to receive the complete user object. [Optional]
- **contributor_details** – If set to True, this parameter enhances the contributors element of the status response to include the screen_name of the contributor. By default only the user_id of the contributor is included. [Optional]
- **include_entities** – The entities node will be disincluded when set to False. [Optional]
- **return_json** (*bool, optional*) – If True JSON data will be returned, instead of twitter.User

Returns A sequence of twitter.Status instances, one for each mention of the user.

GetMutes(*skip_status=False*, *include_entities=False*)

Fetch the sequence of all users (as twitter.User instances), muted by the currently authenticated user.

Parameters

- **skip_status** (*bool, optional*) – If True the statuses will not be returned in the user items.
- **include_entities** (*bool, optional*) – When True, the user entities will be included.

Returns A list of twitter.User instances, one for each muted user.

GetMutesIDs(*stringify_ids=False*)

Fetch the sequence of all user IDs muted by the currently authenticated user.

Parameters **stringify_ids** (*bool, optional*) – If True user IDs will be returned as strings rather than integers.

Returns A list of user IDs for all muted users.

GetMutesIDsPaged(*cursor=-1*, *stringify_ids=False*)

Fetch a page of the user IDs muted by the currently authenticated user.

Parameters

- **cursor** (*int, optional*) – Should be set to -1 if you want the first page, thereafter denotes the page of muted users that you want to return.
- **stringify_ids** (*bool, optional*) – If True user IDs will be returned as strings rather than integers.

Returns next_cursor, previous_cursor, list of user IDs of muted users.

GetMutesPaged(*cursor=-1*, *skip_status=False*, *include_entities=False*)

Fetch a page of the users (as twitter.User instances) muted by the currently authenticated user.

Parameters

- **cursor** (*int, optional*) – Should be set to -1 if you want the first page, thereafter denotes the page of muted users that you want to return.
- **skip_status** (*bool, optional*) – If True the statuses will not be returned in the user items.
- **include_entities** (*bool, optional*) – When True, the user entities will be included.

Returns next_cursor, previous_cursor, list of twitter.User instances, one for each muted user.

GetReplies(*since_id=None*, *count=None*, *max_id=None*, *trim_user=False*)

Get a sequence of status messages representing the 20 most recent replies (status updates prefixed with @twitterID) to the authenticating user.

Parameters

- **since_id** – Returns results with an ID greater than (that is, more recent than) the specified ID. There are limits to the number of Tweets which can be accessed through the API. If the limit of Tweets has occurred since the since_id, the since_id will be forced to the oldest ID available. [Optional]
- **max_id** – Returns results with an ID less than (that is, older than) or equal to the specified ID. [Optional]
- **trim_user** – If True the returned payload will only contain the user IDs, otherwise the payload will contain the full user data item. [Optional]

Returns A sequence of twitter.Status instances, one for each reply to the user.

GetRetweeters(*status_id*, *cursor=None*, *count=100*, *stringify_ids=False*)

Returns a collection of up to 100 user IDs belonging to users who have retweeted the tweet specified by the status_id parameter.

Parameters

- **status_id** – the tweet's numerical ID
- **cursor** – breaks the ids into pages of no more than 100.
- **stringify_ids** – returns the IDs as unicode strings. [Optional]

Returns A list of user IDs

GetRetweets(*statusid*, *count=None*, *trim_user=False*)

Returns up to 100 of the first retweets of the tweet identified by statusid

Parameters

- **statusid** (*int*) – The ID of the tweet for which retweets should be searched for

- **count** (*int, optional*) – The number of status messages to retrieve.
- **trim_user** (*bool, optional*) – If True the returned payload will only contain the user IDs, otherwise the payload will contain the full user data item.

Returns A list of twitter.Status instances, which are retweets of statusid

GetRetweetsOfMe(*count=None*, *since_id=None*, *max_id=None*, *trim_user=False*, *include_entities=True*, *include_user_entities=True*)

Returns up to 100 of the most recent tweets of the user that have been retweeted by others.

Parameters

- **count** – The number of retweets to retrieve, up to 100. Defaults to 20. [Optional]
- **since_id** – Returns results with an ID greater than (newer than) this ID. [Optional]
- **max_id** – Returns results with an ID less than or equal to this ID. [Optional]
- **trim_user** – When True, the user object for each tweet will only be an ID. [Optional]
- **include_entities** – When True, the tweet entities will be included. [Optional]
- **include_user_entities** – When True, the user entities will be included. [Optional]

GetSearch(*term=None*, *raw_query=None*, *geocode=None*, *since_id=None*, *max_id=None*, *until=None*, *since=None*, *count=15*, *lang=None*, *locale=None*, *result_type='mixed'*, *include_entities=None*, *return_json=False*)

Return twitter search results for a given term. You must specify one of term, geocode, or raw_query.

Parameters

- **term** (*str, optional*) – Term to search by. Optional if you include geocode.
- **raw_query** (*str, optional*) – A raw query as a string. This should be everything after the "?" in the URL (i.e., the query parameters). You are responsible for all type checking and ensuring that the query string is properly formatted, as it will only be URL-encoded before be passed directly to Twitter with no other checks performed. For advanced usage only. *This will override any other parameters passed*
- **since_id** (*int, optional*) – Returns results with an ID greater than (that is, more recent than) the specified ID. There are limits to the number of Tweets which can be accessed through the API. If the limit of Tweets has occurred since the since_id, the since_id will be forced to the oldest ID available.
- **max_id** (*int, optional*) – Returns only statuses with an ID less than (that is, older than) or equal to the specified ID.
- **until** (*str, optional*) – Returns tweets generated before the given date. Date should be formatted as YYYY-MM-DD.
- **since** (*str, optional*) – Returns tweets generated since the given date. Date should be formatted as YYYY-MM-DD.
- **geocode** (*str or list or tuple, optional*) – Geolocation within which to search for tweets. Can be either a string in the form of "latitude,longitude,radius" where latitude and longitude are floats and radius is a string such as "1mi" or "1km" ("mi" or "km" are the only units allowed). For example:

```
>>> api.GetSearch(geocode="37.781157,-122.398720,1mi").
```

Otherwise, you can pass a list of either floats or strings for lat/long and a string for radius:

```
>>> api.GetSearch(geocode=[37.781157, -122.398720, "1mi"])
>>> # or:
>>> api.GetSearch(geocode=(37.781157, -122.398720, "1mi"))
>>> # or:
>>> api.GetSearch(geocode=("37.781157", "-122.398720", "1mi"))
```

- **count** (*int, optional*) – Number of results to return. Default is 15 and maxmimum that Twitter returns is 100 irrespective of what you type in.
- **lang** (*str, optional*) – Language for results as ISO 639-1 code. Default is None (all languages).
- **locale** (*str, optional*) – Language of the search query. Currently only 'ja' is effective. This is intended for language-specific consumers and the default should work in the majority of cases.
- **result_type** (*str, optional*) – Type of result which should be returned. Default is "mixed". Valid options are "mixed, "recent", and "popular".
- **include_entities** (*bool, optional*) – If True, each tweet will include a node called "entities". This node offers a variety of metadata about the tweet in a discrete structure, including: user_mentions, urls, and hashtags.
- **return_json** (*bool, optional*) – If True JSON data will be returned, instead of twitter.Userret

Returns A sequence of twitter.Status instances, one for each message containing the term, within the bounds of the geocoded area, or given by the raw_query.

Return type list

GetSentDirectMessages(*since_id=None*, *max_id=None*, *count=None*, *page=None*, *include_entities=True*, *return_json=False*)

Returns a list of the direct messages sent by the authenticating user.

Parameters

- **since_id** (*int, optional*) – Returns results with an ID greater than (that is, more recent than) the specified ID. There are limits to the number of Tweets which can be accessed through the API. If the limit of Tweets has occured since the since_id, the since_id will be forced to the oldest ID available.
- **max_id** (*int, optional*) – Returns results with an ID less than (that is, older than) or equal to the specified ID.
- **count** (*int, optional*) – Specifies the number of direct messages to try and retrieve, up to a maximum of 200. The value of count is best thought of as a limit to the number of Tweets to return because suspended or deleted content is removed after the count has been applied.
- **page** (*int, optional*) – Specifies the page of results to retrieve. Note: there are pagination limits. [Optional]
- **include_entities** (*bool, optional*) – The entities node will be omitted when set to False.
- **return_json** (*bool, optional*) – If True JSON data will be returned, instead of twitter.User

Returns A sequence of twitter.DirectMessage instances

GetShortUrlLength(*https=False*)
Returns number of characters reserved per URL included in a tweet.

Parameters **https** (*bool, optional*) – If True, return number of characters reserved for https urls or, if False, return number of character reserved for http urls.

Returns Number of characters reserved per URL.

Return type (int)

GetStatus(*status_id*, *trim_user=False*, *include_my_retweet=True*, *include_entities=True*, *include_ext_alt_text=True*)
Returns a single status message, specified by the status_id parameter.

Parameters

- **status_id** – The numeric ID of the status you are trying to retrieve.
- **trim_user** – When set to True, each tweet returned in a timeline will include a user object including only the status authors numerical ID. Omit this parameter to receive the complete user object. [Optional]
- **include_my_retweet** – When set to True, any Tweets returned that have been retweeted by the authenticating user will include an additional current_user_retweet node, containing the ID of the source status for the retweet. [Optional]
- **include_entities** – If False, the entities node will be disincluded. This node offers a variety of metadata about the tweet in a discreet structure, including: user_mentions, urls, and hashtags. [Optional]

Returns A twitter.Status instance representing that status message

GetStatusOembed(*status_id=None*, *url=None*, *maxwidth=None*, *hide_media=False*, *hide_thread=False*, *omit_script=False*, *align=None*, *related=None*, *lang=None*)
Returns information allowing the creation of an embedded representation of a Tweet on third party sites.

Specify tweet by the id or url parameter.

Parameters

- **status_id** – The numeric ID of the status you are trying to embed.
- **url** – The url of the status you are trying to embed.
- **maxwidth** – The maximum width in pixels that the embed should be rendered at. This value is constrained to be between 250 and 550 pixels. [Optional]
- **hide_media** – Specifies whether the embedded Tweet should automatically expand images. [Optional]
- **hide_thread** – Specifies whether the embedded Tweet should automatically show the original message in the case that the embedded Tweet is a reply. [Optional]
- **omit_script** – Specifies whether the embedded Tweet HTML should include a <script> element pointing to widgets.js. [Optional]
- **align** – Specifies whether the embedded Tweet should be left aligned, right aligned, or centered in the page. [Optional]
- **related** – A comma sperated string of related screen names. [Optional]
- **lang** – Language code for the rendered embed. [Optional]

Returns A dictionary with the response.

GetStatuses(*status_ids*, *trim_user=False*, *include_entities=True*, *map=False*)
Returns a list of status messages, specified by the status_ids parameter.

Parameters

- **status_ids** – A list of the numeric ID of the statuses you are trying to retrieve.
- **trim_user** – When set to True, each tweet returned in a timeline will include a user object including only the status authors numerical ID. Omit this parameter to receive the complete user object. [Optional]
- **include_entities** – If False, the entities node will be disincluded. This node offers a variety of metadata about the tweet in a discreet structure, including: user_mentions, urls, and hashtags. [Optional]
- **map** – If True, returns a dictionary with status id as key and returned status data (or None if tweet does not exist or is inaccessible) as value. Otherwise returns an unordered list of successfully retrieved Tweets. [Optional]

Returns A dictionary or unordered list (depending on the parameter 'map') of twitter Status instances representing the status messages.

GetStreamFilter(*follow=None*, *track=None*, *locations=None*, *languages=None*, *delimited=None*, *stall_warnings=None*, *filter_level=None*)
Returns a filtered view of public statuses.

Parameters

- **follow** – A list of user IDs to track. [Optional]
- **track** – A list of expressions to track. [Optional]
- **locations** – A list of Longitude,Latitude pairs (as strings) specifying bounding boxes for the tweets' origin. [Optional]
- **delimited** – Specifies a message length. [Optional]
- **stall_warnings** – Set to True to have Twitter deliver stall warnings. [Optional]
- **languages** – A list of Languages. Will only return Tweets that have been detected as being written in the specified languages. [Optional]
- **filter_level** – Specifies level of filtering applied to stream. Set to None, 'low' or 'medium'. [Optional]

Returns A twitter stream

GetStreamSample(*delimited=False*, *stall_warnings=True*)
Returns a small sample of public statuses.

Parameters

- **delimited** – Specifies a message length. [Optional]
- **stall_warnings** – Set to True to have Twitter deliver stall warnings. [Optional]

Returns A Twitter stream

GetSubscriptions(*user_id=None*, *screen_name=None*, *count=20*, *cursor=-1*, *return_json=False*)
Obtain a collection of the lists the specified user is subscribed to. If neither user_id or screen_name is specified, the data returned will be for the authenticated user.

The list will contain a maximum of 20 lists per page by default.

Does not include the user's own lists.

Parameters

- **user_id** (*int, optional*) – The ID of the user for whom to return results for.
- **screen_name** (*str, optional*) – The screen name of the user for whom to return results for.
- **count** (*int, optional*) – The amount of results to return per page. No more than 1000 results will ever be returned in a single page. Defaults to 20.
- **cursor** (*int, optional*) – The "page" value that Twitter will use to start building the list sequence from. Use the value of -1 to start at the beginning. Twitter will return in the result the values for next_cursor and previous_cursor.
- **return_json** (*bool, optional*) – If True JSON data will be returned, instead of twitter.User

Returns A sequence of twitter.List instances, one for each list

Return type twitter.list.List

GetTrendsCurrent(*exclude=None*)
Get the current top trending topics (global)

Parameters **exclude** – Appends the exclude parameter as a request parameter. Currently only exclude=hashtags is supported. [Optional]

Returns A list with 10 entries. Each entry contains a trend.

GetTrendsWoeid(*woeid*, *exclude=None*)
Return the top 10 trending topics for a specific WOEID, if trending information is available for it.

Parameters

- **woeid** – the Yahoo! Where On Earth ID for a location.
- **exclude** – Appends the exclude parameter as a request parameter. Currently only exclude=hashtags is supported. [Optional]

Returns A list with 10 entries. Each entry contains a trend.

GetUser(*user_id=None*, *screen_name=None*, *include_entities=True*, *return_json=False*)
Returns a single user.

Parameters

- **user_id** (*int, optional*) – The id of the user to retrieve.
- **screen_name** (*str, optional*) – The screen name of the user for whom to return results for. Either a user_id or screen_name is required for this method.
- **include_entities** (*bool, optional*) – The entities node will be omitted when set to False.
- **return_json** (*bool, optional*) – If True JSON data will be returned, instead of twitter.User

Returns A twitter.User instance representing that user

GetUserRetweets(*count=None*, *since_id=None*, *max_id=None*, *trim_user=False*)
Fetch the sequence of retweets made by the authenticated user.

Parameters

- **count** – The number of status messages to retrieve. [Optional]
- **since_id** – Returns results with an ID greater than (that is, more recent than) the specified ID. There are limits to the number of Tweets which can be accessed through the API.

If the limit of Tweets has occurred since the since_id, the since_id will be forced to the oldest ID available. [Optional]

- **max_id** – Returns results with an ID less than (that is, older than) or equal to the specified ID. [Optional]
- **trim_user** – If True the returned payload will only contain the user IDs, otherwise the payload will contain the full user data item. [Optional]

Returns A sequence of twitter.Status instances, one for each message up to count

GetUserStream(*replies='all'*, *withuser='user'*, *track=None*, *locations=None*, *delimited=None*, *stall_warnings=None*, *stringify_friend_ids=False*, *filter_level=None*, *session=None*, *include_keepalive=False*)

Returns the data from the user stream.

Parameters

- **replies** – Specifies whether to return additional @replies in the stream. Defaults to 'all'.
- **withuser** – Specifies whether to return information for just the authenticating user, or include messages from accounts the user follows. [Optional]
- **track** – A list of expressions to track. [Optional]
- **locations** – A list of Latitude,Longitude pairs (as strings) specifying bounding boxes for the tweets' origin. [Optional]
- **delimited** – Specifies a message length. [Optional]
- **stall_warnings** – Set to True to have Twitter deliver stall warnings. [Optional]
- **stringify_friend_ids** – Specifies whether to send the friends list preamble as an array of integers or an array of strings. [Optional]
- **filter_level** – Specifies level of filtering applied to stream. Set to None, low or medium. [Optional]

Returns A twitter stream

GetUserSuggestion(*category*)

Returns a list of users in a category :param category: The Category object to limit the search by

Returns A list of users in that category

GetUserSuggestionCategories()

Return the list of suggested user categories, this can be used in GetUserSuggestion function

Returns A list of categories

GetUserTimeline(*user_id=None*, *screen_name=None*, *since_id=None*, *max_id=None*, *count=None*, *include_rts=True*, *trim_user=False*, *exclude_replies=False*)

Fetch the sequence of public Status messages for a single user.

The twitter.Api instance must be authenticated if the user is private.

Parameters

- **user_id** (*int, optional*) – Specifies the ID of the user for whom to return the user_timeline. Helpful for disambiguating when a valid user ID is also a valid screen name.

- **screen_name** (*str, optional*) – Specifies the screen name of the user for whom to return the user_timeline. Helpful for disambiguating when a valid screen name is also a user ID.
- **since_id** (*int, optional*) – Returns results with an ID greater than (that is, more recent than) the specified ID. There are limits to the number of Tweets which can be accessed through the API. If the limit of Tweets has occurred since the since_id, the since_id will be forced to the oldest ID available.
- **max_id** (*int, optional*) – Returns only statuses with an ID less than (that is, older than) or equal to the specified ID.
- **count** (*int, optional*) – Specifies the number of statuses to retrieve. May not be greater than 200.
- **include_rts** (*bool, optional*) – If True, the timeline will contain native retweets (if they exist) in addition to the standard stream of tweets.
- **trim_user** (*bool, optional*) – If True, statuses will only contain the numerical user ID only. Otherwise a full user object will be returned for each status.
- **exclude_replies** (*bool, optional*) – If True, this will prevent replies from appearing in the returned timeline. Using exclude_replies with the count parameter will mean you will receive up-to count tweets - this is because the count parameter retrieves that many tweets before filtering out retweets and replies. This parameter is only supported for JSON and XML responses.

Returns A sequence of Status instances, one for each message up to count

GetUsersSearch(*term=None*, *page=1*, *count=20*, *include_entities=None*)
Return twitter user search results for a given term.

Parameters

- **term** – Term to search by.
- **page** – Page of results to return. Default is 1 [Optional]
- **count** – Number of results to return. Default is 20 [Optional]
- **include_entities** – If True, each tweet will include a node called "entities,". This node offers a variety of metadata about the tweet in a discrete structure, including: user_mentions, urls, and hashtags. [Optional]

Returns A sequence of twitter.User instances, one for each message containing the term

IncomingFriendship(*cursor=None*, *stringify_ids=None*)
Returns a collection of user IDs belonging to users who have pending request to follow the authenticated user.

Parameters

- **cursor** – breaks the ids into pages of no more than 5000.
- **stringify_ids** – returns the IDs as unicode strings. [Optional]

Returns A list of user IDs

InitializeRateLimit()
Make a call to the Twitter API to get the rate limit status for the currently authenticated user or application.

Returns None.

LookupFriendship(*user_id=None*, *screen_name=None*, *return_json=False*)
Lookup friendship status for user to authed user.

Users may be specified either as lists of either user_ids, screen_names, or twitter.User objects. The list of users that are queried is the union of all specified parameters.

Up to 100 users may be specified.

Parameters

- **user_id** (*int, User, or list of ints or Users, optional*) – A list of user_ids to retrieve extended information.
- **screen_name** (*string, User, or list of strings or Users, optional*) – A list of screen_names to retrieve extended information.
- **return_json** (*bool, optional*) – If True JSON data will be returned, instead of twitter.User

Returns A list of twitter.UserStatus instance representing the friendship status between the specified users and the authenticated user.

Return type list

OutgoingFriendship(*cursor=None*, *stringify_ids=None*)

Returns a collection of user IDs for every protected user for whom the authenticated user has a pending follow request.

Parameters

- **cursor** – breaks the ids into pages of no more than 5000.
- **stringify_ids** – returns the IDs as unicode strings. [Optional]

Returns A list of user IDs

PostDirectMessage(*text*, *user_id=None*, *screen_name=None*, *return_json=False*)

Post a twitter direct message from the authenticated user.

Parameters

- **text** – The message text to be posted.
- **user_id** – The ID of the user who should receive the direct message.
- **return_json** (*bool, optional*) – If True JSON data will be returned, instead of twitter.DirectMessage

Returns A twitter.DirectMessage instance representing the message posted

PostMediaMetadata(*media_id*, *alt_text=None*)

Provide addtional data for uploaded media.

Parameters

- **media_id** – ID of a previously uploaded media item.
- **alt_text** – Image Alternate Text.

PostRetweet(*status_id*, *trim_user=False*)

Retweet a tweet with the Retweet API.

Parameters

- **status_id** – The numerical id of the tweet that will be retweeted
- **trim_user** – If True the returned payload will only contain the user IDs, otherwise the payload will contain the full user data item. [Optional]

Returns A twitter.Status instance representing the original tweet with retweet details embedded.

PostUpdate(*status*, *media=None*, *media_additional_owners=None*, *media_category=None*, *in_reply_to_status_id=None*, *auto_populate_reply_metadata=False*, *exclude_reply_user_ids=None*, *latitude=None*, *longitude=None*, *place_id=None*, *display_coordinates=False*, *trim_user=False*, *verify_status_length=True*, *attachment_url=None*)

Post a twitter status message from the authenticated user.

https://dev.twitter.com/docs/api/1.1/post/statuses/update

Parameters

- **status** (*str*) – The message text to be posted. Must be less than or equal to CHARACTER_LIMIT characters.
- **media** (*int, str, fp, optional*) – A URL, a local file, or a file-like object (something with a read() method), or a list of any combination of the above.
- **media_additional_owners** (*list, optional*) – A list of user ids representing Twitter users that should be able to use the uploaded media in their tweets. If you pass a list of media, then additional_owners will apply to each object. If you need more granular control, please use the UploadMedia* methods.
- **media_category** (*str, optional*) – Only for use with the AdsAPI. See https://dev.twitter.com/ads/creative/promoted-video-overview if this applies to your application.
- **in_reply_to_status_id** (*int, optional*) – The ID of an existing status that the status to be posted is in reply to. This implicitly sets the in_reply_to_user_id attribute of the resulting status to the user ID of the message being replied to. Invalid/missing status IDs will be ignored.
- **auto_populate_reply_metadata** (*bool, optional*) – Automatically include the @usernames of the users mentioned or participating in the tweet to which this tweet is in reply.
- **exclude_reply_user_ids** (*list, optional*) – Remove given user_ids (*not* @usernames) from the tweet's automatically generated reply metadata.
- **attachment_url** (*str, optional*) – URL to an attachment resource: one to four photos, a GIF, video, Quote Tweet, or DM deep link. If not specified and media parameter is not None, we will attach the first media object as the attachment URL. If a bad URL is passed, Twitter will raise an error.
- **latitude** (*float, optional*) – Latitude coordinate of the tweet in degrees. Will only work in conjunction with longitude argument. Both longitude and latitude will be ignored by twitter if the user has a false geo_enabled setting.
- **longitude** (*float, optional*) – Longitude coordinate of the tweet in degrees. Will only work in conjunction with latitude argument. Both longitude and latitude will be ignored by twitter if the user has a false geo_enabled setting.
- **place_id** (*int, optional*) – A place in the world. These IDs can be retrieved from GET geo/reverse_geocode.
- **display_coordinates** (*bool, optional*) – Whether or not to put a pin on the exact coordinates a tweet has been sent from.
- **trim_user** (*bool, optional*) – If True the returned payload will only contain the user IDs, otherwise the payload will contain the full user data item.
- **verify_status_length** (*bool, optional*) – If True, api throws a hard error that the status is over CHARACTER_LIMIT characters. If False, Api will attempt to post the status.

Returns (twitter.Status) A twitter.Status instance representing the message posted.

PostUpdates(*status*, *continuation=None*, ***kwargs*)
Post one or more twitter status messages from the authenticated user.

Unlike api.PostUpdate, this method will post multiple status updates if the message is longer than CHARACTER_LIMIT characters.

Parameters

- **status** – The message text to be posted. May be longer than CHARACTER_LIMIT characters.
- **continuation** – The character string, if any, to be appended to all but the last message. Note that Twitter strips trailing '...' strings from messages. Consider using the unicode u2026 character (horizontal ellipsis) instead. [Defaults to None]
- ****kwargs** – See api.PostUpdate for a list of accepted parameters.

Returns A of list twitter.Status instance representing the messages posted.

SetCache(*cache*)
Override the default cache. Set to None to prevent caching.

Parameters **cache** – An instance that supports the same API as the twitter._FileCache

SetCacheTimeout(*cache_timeout*)
Override the default cache timeout.

Parameters **cache_timeout** – Time, in seconds, that responses should be reused.

SetCredentials(*consumer_key*, *consumer_secret*, *access_token_key=None*, *access_token_secret=None*, *application_only_auth=False*)
Set the consumer_key and consumer_secret for this instance

Parameters

- **consumer_key** – The consumer_key of the twitter account.
- **consumer_secret** – The consumer_secret for the twitter account.
- **access_token_key** – The oAuth access token key value you retrieved from running get_access_token.py.
- **access_token_secret** – The oAuth access token's secret, also retrieved from the get_access_token.py run.
- **application_only_auth** – Whether to generate a bearer token and use Application-Only Auth

SetSource(*source*)
Suggest the "from source" value to be displayed on the Twitter web site.

The value of the 'source' parameter must be first recognized by the Twitter server.

New source values are authorized on a case by case basis by the Twitter development team.

Parameters **source** – The source name as a string. Will be sent to the server as the 'source' parameter.

SetUrllib(*urllib*)
Override the default urllib implementation.

Parameters **urllib** – An instance that supports the same API as the urllib2 module

SetUserAgent(*user_agent*)
Override the default user agent.

Parameters **user_agent** – A string that should be send to the server as the user-agent.

SetXTwitterHeaders(*client*, *url*, *version*)
Set the X-Twitter HTTP headers that will be sent to the server.

Parameters

- **client** – The client name as a string. Will be sent to the server as the 'X-Twitter-Client' header.
- **url** – The URL of the meta.xml as a string. Will be sent to the server as the 'X-Twitter-Client-URL' header.
- **version** – The client version as a string. Will be sent to the server as the 'X-Twitter-Client-Version' header.

ShowFriendship(*source_user_id=None*, *source_screen_name=None*, *target_user_id=None*, *target_screen_name=None*)
Returns information about the relationship between the two users.

Parameters

- **source_id** – The user_id of the subject user [Optional]
- **source_screen_name** – The screen_name of the subject user [Optional]
- **target_id** – The user_id of the target user [Optional]
- **target_screen_name** – The screen_name of the target user [Optional]

Returns A Twitter Json structure.

ShowSubscription(*owner_screen_name=None*, *owner_id=None*, *list_id=None*, *slug=None*, *user_id=None*, *screen_name=None*, *include_entities=False*, *skip_status=False*, *return_json=False*)
Check if the specified user is a subscriber of the specified list.

Returns the user if they are subscriber.

Parameters

- **owner_screen_name** (*str, optional*) – The screen_name of the user who owns the list being requested by a slug.
- **owner_id** (*int, optional*) – The user ID of the user who owns the list being requested by a slug.
- **list_id** (*int, optional*) – The numerical ID of the list.
- **slug** (*str, optional*) – You can identify a list by its slug instead of its numerical ID. If you decide to do so, note that you'll also have to specify the list owner using the owner_id or owner_screen_name parameters.
- **user_id** (*int, optional*) – The user_id or a list of user_id's to add to the list. If not given, then screen_name is required.
- **screen_name** (*str, optional*) – The screen_name or a list of screen_name's to add to the list. If not given, then user_id is required.
- **include_entities** (*bool, optional*) – If False, the timeline will not contain additional metadata. Defaults to True.
- **skip_status** (*bool, optional*) – If True the statuses will not be returned in the user items.

- **return_json** (*bool, optional*) – If True JSON data will be returned, instead of twitter.User

Returns A twitter.User instance representing the user requested.

Return type twitter.user.User

UpdateBanner(*image*, *include_entities=False*, *skip_status=False*)
Updates the authenticated users profile banner.

Parameters

- **image** – Location of image in file system
- **include_entities** – If True, each tweet will include a node called "entities." This node offers a variety of metadata about the tweet in a discrete structure, including: user_mentions, urls, and hashtags. [Optional]

Returns A twitter.List instance representing the list subscribed to

UpdateFriendship(*user_id=None*, *screen_name=None*, *follow=True*, *retweets=True*, ***kwargs*)
Updates a friendship with the user specified by the user_id or screen_name.

Parameters

- **user_id** (*int, optional*) – A user_id to update
- **screen_name** (*str, optional*) – A screen_name to update
- **follow** (*bool, optional*) – Set to False to disable notifications for the target user
- **retweets** (*bool, optional*) – Enable or disable retweets from the target user.
- **device** – Set to False to disable notifications for the target user

Returns A twitter.User instance representing the befriended user.

UpdateImage(*image*, *include_entities=False*, *skip_status=False*)
Update a User's profile image. Change may not be immediately reflected due to image processing on Twitter's side.

Parameters

- **image** (*str*) – Location of local image file to use.
- **include_entities** (*bool, optional*) – Include the entities node in the return data.
- **skip_status** (*bool, optional*) – Include the User's last Status in the User entity returned.

Returns Updated User object.

Return type (*twitter.models.User*)

UpdateProfile(*name=None*, *profileURL=None*, *location=None*, *description=None*, *profile_link_color=None*, *include_entities=False*, *skip_status=False*)
Update's the authenticated user's profile data.

Parameters

- **name** (*str, optional*) – Full name associated with the profile.
- **profileURL** (*str, optional*) – URL associated with the profile. Will be prepended with "http://" if not present.
- **location** (*str, optional*) – The city or country describing where the user of the account is located. The contents are not normalized or geocoded in any way.

- **description** (*str, optional*) – A description of the user owning the account.
- **profile_link_color** (*str, optional*) – hex value of profile color theme. formated without '#' or '0x'. Ex: FF00FF
- **include_entities** (*bool, optional*) – The entities node will be omitted when set to False.
- **skip_status** (*bool, optional*) – When set to either True, t or 1 then statuses will not be included in the returned user objects.

Returns A twitter.User instance representing the modified user.

UploadMediaChunked(*media*, *additional_owners=None*, *media_category=None*)

Upload a media file to Twitter in multiple requests.

Parameters

- **media** – File-like object to upload.
- **additional_owners** – additional Twitter users that are allowed to use The uploaded media. Should be a list of integers. Maximum number of additional owners is capped at 100 by Twitter.
- **media_category** – Category with which to identify media upload. Only use with Ads API & video files.

Returns ID of the uploaded media returned by the Twitter API. Raises if unsuccesful.

Return type media_id

UploadMediaSimple(*media*, *additional_owners=None*, *media_category=None*)

Upload a media file to Twitter in one request. Used for small file uploads that do not require chunked uploads.

Parameters

- **media** – File-like object to upload.
- **additional_owners** – additional Twitter users that are allowed to use The uploaded media. Should be a list of integers. Maximum number of additional owners is capped at 100 by Twitter.
- **media_category** – Category with which to identify media upload. Only use with Ads API & video files.

Returns ID of the uploaded media returned by the Twitter API or 0.

Return type media_id

UsersLookup(*user_id=None*, *screen_name=None*, *users=None*, *include_entities=True*, *return_json=False*)

Fetch extended information for the specified users.

Users may be specified either as lists of either user_ids, screen_names, or twitter.User objects. The list of users that are queried is the union of all specified parameters.

No more than 100 users may be given per request.

Parameters

- **user_id** (*int, list, optional*) – A list of user_ids to retrieve extended information.
- **screen_name** (*str, list, optional*) – A list of screen_names to retrieve extended information.

- **users** (*list, optional*) – A list of twitter.User objects to retrieve extended information.
- **include_entities** (*bool, optional*) – The entities node that may appear within embedded statuses will be excluded when set to False.
- **return_json** (*bool, optional*) – If True JSON data will be returned, instead of twitter.User

Returns A list of twitter.User objects for the requested users

VerifyCredentials(*include_entities=None*, *skip_status=None*, *include_email=None*)
Returns a twitter.User instance if the authenticating user is valid.

Parameters

- **include_entities** – Specifies whether to return additional @replies in the stream.
- **skip_status** – When set to either true, t or 1 statuses will not be included in the returned user object.
- **include_email** – Use of this parameter requires whitelisting. When set to true email will be returned in the user objects as a string. If the user does not have an email address on their account, or if the email address is un-verified, null will be returned. If your app is not whitelisted, then the 'email' key will not be present in the json response.

Returns A twitter.User instance representing that user if the credentials are valid, None otherwise.

exception twitter.error.**PythonTwitterDeprecationWarning**
Bases: exceptions.DeprecationWarning

Base class for python-twitter deprecation warnings

exception twitter.error.**PythonTwitterDeprecationWarning330**
Bases: *twitter.error.PythonTwitterDeprecationWarning*

Warning for features to be removed in version 3.3.0

exception twitter.error.**PythonTwitterDeprecationWarning340**
Bases: *twitter.error.PythonTwitterDeprecationWarning*

Warning for features to be removed in version 3.4.0

exception twitter.error.**TwitterError**
Bases: exceptions.Exception

Base class for Twitter errors

message
Returns the first argument used to construct this error.

10.2 Models

class twitter.models.**Category**(***kwargs*)
Bases: *twitter.models.TwitterModel*

A class representing the suggested user category structure.

class twitter.models.**DirectMessage**(***kwargs*)
Bases: *twitter.models.TwitterModel*

A class representing a Direct Message.

class twitter.models.**Hashtag**(***kwargs*)
Bases: *twitter.models.TwitterModel*

A class representing a twitter hashtag.

class twitter.models.**List**(***kwargs*)
Bases: *twitter.models.TwitterModel*

A class representing the List structure used by the twitter API.

class twitter.models.**Media**(***kwargs*)
Bases: *twitter.models.TwitterModel*

A class representing the Media component of a tweet.

class twitter.models.**Status**(***kwargs*)
Bases: *twitter.models.TwitterModel*

A class representing the Status structure used by the twitter API.

classmethod NewFromJsonDict(*data*, ***kwargs*)
Create a new instance based on a JSON dict.

Parameters **data** – A JSON dict, as converted from the JSON in the twitter API

Returns A twitter.Status instance

created_at_in_seconds
Get the time this status message was posted, in seconds since the epoch (1 Jan 1970).

Returns The time this status message was posted, in seconds since the epoch.

Return type int

class twitter.models.**Trend**(***kwargs*)
Bases: *twitter.models.TwitterModel*

A class representing a trending topic.

volume

class twitter.models.**TwitterModel**(***kwargs*)
Bases: object

Base class from which all twitter models will inherit.

AsDict()
Create a dictionary representation of the object. Please see inline comments on construction when dictionaries contain TwitterModels.

AsJsonString(*ensure_ascii=True*)
Returns the TwitterModel as a JSON string based on key/value pairs returned from the AsDict() method.

classmethod NewFromJsonDict(*data*, ***kwargs*)
Create a new instance based on a JSON dict. Any kwargs should be supplied by the inherited, calling class.

Parameters **data** – A JSON dict, as converted from the JSON in the twitter API.

class twitter.models.**Url**(***kwargs*)
Bases: *twitter.models.TwitterModel*

A class representing an URL contained in a tweet.

class twitter.models.**User**(***kwargs*)
Bases: *twitter.models.TwitterModel*

A class representing the User structure.

classmethod NewFromJsonDict(*data*, ***kwargs*)

class twitter.models.**UserStatus**(***kwargs*)
Bases: *twitter.models.TwitterModel*

A class representing the UserStatus structure. This is an abbreviated form of the twitter.User object.

connections

class twitter.ratelimit.**EndpointRateLimit**(*limit*, *remaining*, *reset*)
Bases: tuple

limit
Alias for field number 0

remaining
Alias for field number 1

reset
Alias for field number 2

class twitter.ratelimit.**RateLimit**(***kwargs*)
Bases: object

Object to hold the rate limit status of various endpoints for the twitter.Api object.

This object is generally attached to the API as Api.rate_limit, but is not created until the user makes a method call that uses _RequestUrl() or calls Api.InitializeRateLimit(), after which it get created and populated with rate limit data from Twitter.

Calling Api.InitializeRateLimit() populates the object with all of the rate limits for the endpoints defined by Twitter; more info is available here:

https://dev.twitter.com/rest/public/rate-limits

https://dev.twitter.com/rest/public/rate-limiting

https://dev.twitter.com/rest/reference/get/application/rate_limit_status

Once a resource (i.e., an endpoint) has been requested, Twitter's response will contain the current rate limit status as part of the headers, i.e.:

```
x-rate-limit-limit
x-rate-limit-remaining
x-rate-limit-reset
```

`limit` is the generic limit for that endpoint, `remaining` is how many more times you can make a call to that endpoint, and `reset` is the time (in seconds since the epoch) until remaining resets to its default for that endpoint.

Generally speaking, each endpoint has a 15-minute reset time and endpoints can either make 180 or 15 requests per window. According to Twitter, any endpoint not defined in the rate limit chart or the response from a GET request to `application/rate_limit_status.json` should be assumed to be 15 requests per 15 minutes.

get_limit(*url*)
Gets a EndpointRateLimit object for the given url.

Parameters **url** (*str, optional*) – URL of the endpoint for which to return the rate limit status.

Returns EndpointRateLimit object containing rate limit information.

Return type namedtuple

set_limit(*url*, *limit*, *remaining*, *reset*)
If a resource family is unknown, add it to the object's dictionary. This is to deal with new endpoints being added to the API, but not necessarily to the information returned by `/account/rate_limit_status.json` endpoint.

For example, if Twitter were to add an endpoint `/puppies/lookup.json`, the RateLimit object would create a resource family `puppies` and add `/puppies/lookup` as the endpoint, along with whatever limit, remaining hits available, and reset time would be applicable to that resource+endpoint pair.

Parameters

- **url** (*str*) – URL of the endpoint being fetched.
- **limit** (*int*) – Max number of times a user or app can hit the endpoint before being rate limited.
- **remaining** (*int*) – Number of times a user or app can access the endpoint before being rate limited.
- **reset** (*int*) – Epoch time at which the rate limit window will reset.

set_unknown_limit(*url*, *limit*, *remaining*, *reset*)

static url_to_resource(*url*)
Take a fully qualified URL and attempts to return the rate limit resource family corresponding to it. For example:

```
>>> RateLimit.url_to_resource('https://api.twitter.com/1.1/statuses/lookup.
↪json?id=317')
>>> '/statuses/lookup'
```

Parameters **url** (*str*) – URL to convert to a resource family.

Returns Resource family corresponding to the URL.

Return type string

class twitter.ratelimit.**ResourceEndpoint**(*regex*, *resource*)
Bases: tuple

regex
Alias for field number 0

resource
Alias for field number 1

10.3 Utilities

twitter.twitter_utils.**calc_expected_status_length**(*status*, *short_url_length=23*)
Calculates the length of a tweet, taking into account Twitter's replacement of URLs with https://t.co links.

Parameters

- **status** – text of the status message to be posted.
- **short_url_length** – the current published https://t.co links

Returns Expected length of the status message as an integer.

twitter.twitter_utils.**enf_type**(*field*, *_type*, *val*)
Checks to see if a given val for a field (i.e., the name of the field) is of the proper _type. If it is not, raises a TwitterError with a brief explanation.

Parameters

- **field** – Name of the field you are checking.
- **_type** – Type that the value should be returned as.
- **val** – Value to convert to _type.

Returns val converted to type _type.

twitter.twitter_utils.**http_to_file**(*http*)

twitter.twitter_utils.**is_url**(*text*)
Checks to see if a bit of text is a URL.

Parameters **text** – text to check.

Returns Boolean of whether the text should be treated as a URL or not.

twitter.twitter_utils.**parse_arg_list**(*args*, *attr*)

twitter.twitter_utils.**parse_media_file**(*passed_media*, *async_upload=False*)
Parses a media file and attempts to return a file-like object and information about the media file.

Parameters

- **passed_media** – media file which to parse.
- **async_upload** – flag, for validation media file attributes.

Returns file-like object, the filename of the media file, the file size, and the type of media.

CHAPTER 11

Introduction

This library provides a pure Python interface for the Twitter API. It works with Python 2.7+ and Python 3.

Twitter provides a service that allows people to connect via the web, IM, and SMS. Twitter exposes a web services API and this library is intended to make it even easier for Python programmers to use.

CHAPTER 12

Indices and tables

- genindex
- modindex
- search

Python Module Index

t

Index

H

I

L

M

N

O

P

R

S

T

U

V