

---

# **python-twitter Documentation**

***Release 3.4.2***

**python-twitter@googlegroups.com**

**Dec 19, 2021**



---

## Contents

---

<b>1</b>	<b>Installation &amp; Testing</b>	<b>3</b>
<b>2</b>	<b>Getting Started</b>	<b>5</b>
<b>3</b>	<b>Contributing</b>	<b>11</b>
<b>4</b>	<b>Migration from v2 to v3</b>	<b>13</b>
<b>5</b>	<b>Changelog</b>	<b>19</b>
<b>6</b>	<b>Rate Limiting</b>	<b>23</b>
<b>7</b>	<b>Models</b>	<b>25</b>
<b>8</b>	<b>Searching</b>	<b>27</b>
<b>9</b>	<b>Using with Django</b>	<b>29</b>
<b>10</b>	<b>Modules Documentation</b>	<b>31</b>
<b>11</b>	<b>Introduction</b>	<b>33</b>
<b>12</b>	<b>Indices and tables</b>	<b>35</b>



**A Python wrapper around the Twitter API.**

Author: The Python-Twitter Developers <[python-twitter@googlegroups.com](mailto:python-twitter@googlegroups.com)>

Contents:



### 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
```



## 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.

## Create an application

### Application Details

**Name \***

Your application name. This is used to attribute the source of a tweet and in user-facing authorization screens. 32 characters max.

**Description \***

Your application description, which will be shown in user-facing authorization screens. Between 10 and 200 characters max.

**Website \***

Your application's publicly accessible home page, where users can go to download, make use of, or find out more information about your application. This fully-qualified URL is used in the source attribution for tweets created by your application and will be shown in user-facing authorization screens. (If you don't have a URL yet, just put a placeholder here but remember to change it later.)

**Callback URL**

Where should we return after successfully authenticating? OAuth 1.0a applications should explicitly specify their oauth\_callback URL on the request token step, regardless of the value given here. To restrict your application from using callbacks, leave this field blank.

### Developer Agreement

Effective: May 18, 2015.

This Twitter Developer Agreement ("**Agreement**") is made between you (either an individual or an entity, referred to herein as "**you**") and Twitter, Inc. and Twitter International Company (collectively, "**Twitter**") and governs your access to and use of the Licensed Material (as defined below).

PLEASE READ THE TERMS AND CONDITIONS OF THIS AGREEMENT CAREFULLY, INCLUDING WITHOUT LIMITATION ANY LINKED TERMS AND CONDITIONS APPEARING OR REFERENCED BELOW, WHICH ARE HEREBY MADE PART OF THIS LICENSE AGREEMENT. BY USING THE LICENSED MATERIAL, YOU ARE AGREEING THAT YOU HAVE READ, AND THAT YOU AGREE TO COMPLY WITH AND TO BE BOUND BY THE TERMS AND CONDITIONS OF THIS AGREEMENT AND ALL APPLICABLE LAWS AND REGULATIONS IN THEIR ENTIRETY WITHOUT LIMITATION OR QUALIFICATION. IF YOU DO NOT AGREE TO BE BOUND BY THIS AGREEMENT, THEN YOU MAY NOT ACCESS OR OTHERWISE USE THE LICENSED MATERIAL. THIS AGREEMENT IS EFFECTIVE AS OF THE FIRST DATE THAT YOU USE THE LICENSED MATERIAL ("**EFFECTIVE DATE**").

IF YOU ARE AN INDIVIDUAL REPRESENTING AN ENTITY, YOU ACKNOWLEDGE THAT YOU HAVE THE APPROPRIATE AUTHORITY TO ACCEPT THIS AGREEMENT ON BEHALF OF SUCH ENTITY. YOU MAY NOT USE THE LICENSED MATERIAL AND MAY NOT ACCEPT THIS AGREEMENT IF YOU ARE NOT OF LEGAL AGE TO FORM A BINDING CONTRACT WITH TWITTER, OR

☒ Yes, I agree

### Having trouble creating your application?

If you're having trouble fulfilling application creation requirements, please contact our Platform Operations team by using the "I have an API policy question not covered by these points" option of the contact form at <https://support.twitter.com/forms/platform>

Create your Twitter application

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.

Apps > [python-twitter-reader-fut](#)


---

[App details](#)   [Keys and tokens](#)   [Permissions](#)

---

**App details**  
Details and URLs

Edit ▾

 **App icon**  
App icon is default, click edit to upload.

**App Name**  
python-twit[REDACTED]

**Description**  
Test python twitter application that reads tweets.

**Website URL**  
[REDACTED]

**Sign in with Twitter**  
Disabled

**Callback URL**  
<https://127.0.0.1:5000>

**Terms of service URL**  
None

**Privacy policy URL**  
None

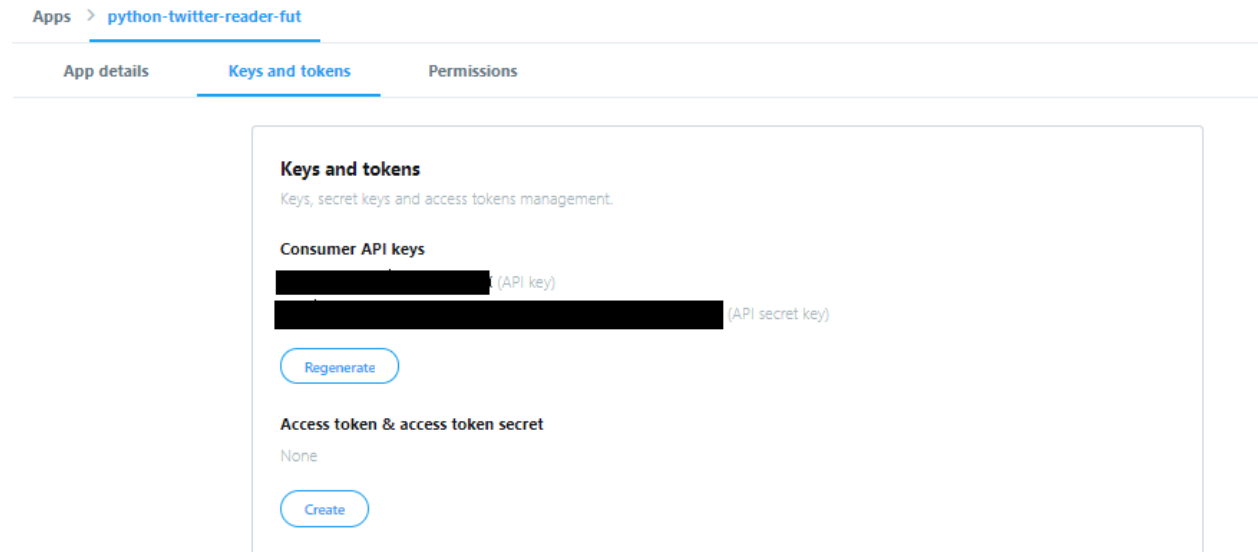
**Organization name**  
None

**Organization website URL**  
None

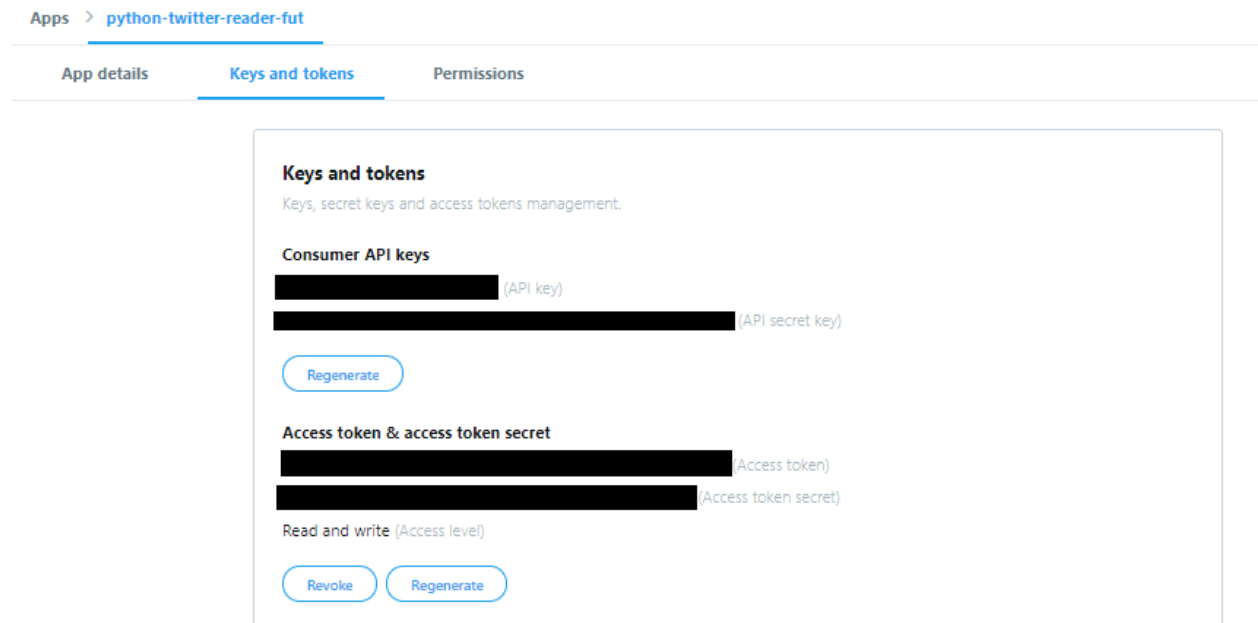
**App usage**  
Testing a python program that reads tweets and store them for later creating a sentiment analysys machine learning model.

### 2.1.3 Your Keys

Click on the “Keys and Access Tokens” tab on the top.



Under the “Access token & access token secret” option, click on the “create” button to generate a new access token and token secret.



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 your 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.



### 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](#).





### 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 use `twitter.api.Api.GetListMembersPaged` instead.

#### 4.1.12 `twitter.api.Api.GetStatus()`

- Kwargs `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()`

- Kwargs `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()`

- Kwargs `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.



### 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 deprecated 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 instantiating 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 occurring 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.

### 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.

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`



## 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**

#### **10.2 Models**

#### **10.3 Utilities**



# 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`