

---

# twtxt Documentation

*Release 1.2.2*

**bucket**

**Sep 26, 2017**



---

## Contents

---

|          |                            |           |
|----------|----------------------------|-----------|
| <b>1</b> | <b>User Guide</b>          | <b>3</b>  |
| 1.1      | Introduction . . . . .     | 3         |
| 1.2      | Installation . . . . .     | 3         |
| 1.3      | Quickstart . . . . .       | 4         |
| 1.4      | Usage . . . . .            | 5         |
| 1.5      | Configuration . . . . .    | 6         |
| 1.6      | twtxt file . . . . .       | 8         |
| 1.7      | Registry . . . . .         | 8         |
| 1.8      | Discoverability . . . . .  | 10        |
| <b>2</b> | <b>Community</b>           | <b>13</b> |
| <b>3</b> | <b>API Reference</b>       | <b>15</b> |
| 3.1      | API . . . . .              | 15        |
|          | <b>Python Module Index</b> | <b>19</b> |



Release: v1.2.2.

Welcome to twtxt's documentation. This documentation is divided into multiple parts. We recommend you to get started with the *Installation* and then head over to the *Quickstart* section. If you don't know what this is about read the *Introduction* first. There is a more detailed *Usage* section about how to use twtxt via the CLI. The internals of twtxt are documented in the *API* chapter.

Feel free to contribute to this project. The source code is maintained on [GitHub](#).



## Introduction

**twtxt** is a decentralised, minimalist microblogging service for hackers.

You want to get some thoughts out on the internet in a convenient and slick way while also following the gibberish of others? Instead of signing up at a closed and/or regulated microblogging platform, getting your status updates out with twtxt is as easy as putting them in a publicly accessible text file. The URL pointing to this file is your identity, your account. twtxt then tracks these text files, like a feedreader, and builds your unique timeline out of them, depending on which files you track. The format is simple, human readable, and integrates well with UNIX command line utilities.

**tl;dr:** twtxt is a CLI tool, as well as a format specification for self-hosted flat file based microblogging.

## Demonstration

### Features

- A beautiful command-line interface thanks to click.
- Asynchronous HTTP requests thanks to asyncio/aiohttp and Python 3.
- Integrates well with existing tools (scp, cut, echo, date, etc.) and your shell.
- Don't like the official client? Tweet using `echo -e "`date -Im`\tHello world!" >> twtxt.txt!`

## Installation

The following sections describe how to install twtxt in different ways on your machine. Currently the we support Windows, Mac OS X and Linux via [pip](#).

**Requirements:**

- Python `>= 3.4.1`
- Recent version of `pip`

## Release version

Install twtxt using `pip`:

```
$ pip3 install twtxt
```

---

**Note:** Instead of installing the package globally (as root), you may want to install this package locally by passing `--user` to `pip`, make sure that you append `~/ .local/bin/` to your `$PATH`. Using `pyvenv` and running twtxt from within a virtualenv is also an option!

---

Packages exist for the following systems:

- Arch Linux (AUR)
- Mac OS X (homebrew)

## Development version

Clone the `git` repository:

```
$ git clone https://github.com/bucket/twtxt.git
```

We recommend you to develop inside a virtualenv:

```
$ pyvenv env
...
$ source env/bin/activate
```

Install the package via `pip` in developer mode:

```
$ pip3 install -e twtxt/[dev]
```

---

**Note:** Appending `[dev]` to the package name or target location will also install the packages required for testing twtxt, by making use of `setuptools's extra` functionality.

---

## Quickstart

Use twtxt's **quickstart** command to bootstrap a default configuration:

```
$ twtxt quickstart

twtxt - quickstart
=====

This wizard will generate a basic configuration file for twtxt with all
mandatory options set. You can change all of these later with either twtxt
```



itself or by editing the config file manually. Have a look at the docs to get information about the other available options and their meaning.

```
Please enter your desired nick [$USER]: <NICKNAME>
Please enter the desired location for your config
  file [ ~/.config/twtxt/config ]: <CONFIG FILE LOCATION>
Please enter the desired location for your twtxt
  file [ ~/twtxt.txt ]: <TWTEXTX FILE LOCATION>
Please enter the URL your twtxt file will be accessible
  from [ https://example.org/twtxt.txt ]: <TWTEXT URL>
Do you want to disclose your identity?
  Your nick and URL will be shared when making HTTP requests [y/N]: y

Do you want to follow the twtxt news feed? [Y/n]: y

✓ Created config file at '~/.config/twtxt/config'.
✓ Created twtxt file at '~/twtxt.txt'.
```

The quickstart wizard prompts for the following configuration values:

- Your desired nick name, doesn't need to be unique (<NICKNAME>)
- The desired location for your config file (<CONFIG FILE LOCATION>)
- The desired location for your twtxt file (<TWTEXTX FILE LOCATION>)
- The URL your twtxt file will be accessible from remotely (<TWTEXT URL>)
- If you want to disclose your identity. If True your nick and URL will be used in the User-Agent header attribute when fetching other twtxt files via HTTP, see [Discoverability](#).
- If you want to follow the official twtxt news feed

The configurations can easily be changed in the twtxt configuration file. See [Configuration](#).

## Usage

twtxt features an excellent command-line interface thanks to [click](#). Don't hesitate to append `--help` or call commands without arguments to get information about all available commands, options and arguments.

Here are a few of the most common operations you may encounter when using twtxt:

### Follow a source

```
$ twtxt follow bob http://bobsplace.xyz/twtxt
✓ You're now following bob.
```

### List all sources you're following

```
$ twtxt following
alice @ https://example.org/alice.txt
bob @ http://bobsplace.xyz/twtxt
```

## Unfollow a source

```
$ twtxt unfollow bob
✓ You've unfollowed bob.
```

## Post a status update

```
$ twtxt tweet "Hello, this is twtxt!"
```

## View your timeline

```
$ twtxt timeline

bob (5 minutes ago):
This is my first "tweet". :)

alice (2 hours ago):
I wonder if this is a thing?
```

## View feed of specific source

```
$ twtxt view twtxt

twtxt (a day ago):
Fiat Lux!
```

```
$ twtxt view http://example.org/twtxt.txt

http://example.org/twtxt.txt (a day ago):
Fiat Lux!
```

## Edit twtxt configuration

```
$ twtxt config twtxt.nick tuxtime
$ twtxt config twtxt.nick
tuxtime
$ twtxt config --remove twtxt.nick
$ twtxt config --edit
# opens your sensible-editor to edit the config file
```

## Configuration

twtxt uses a simple INI-like configuration file. It's recommended to use `twtxt quickstart` to create it. On Linux twtxt checks `~/.config/twtxt/config` for its configuration. On OSX it uses `~/Library/Application Support/twtxt/config`. Consult `click.get_app_dir()` to find out the config directory for other operating systems.

Here's an example conf file, showing every currently supported option:

```
[twtxt]
nick = buckket
twtfilfile = ~/twtxt.txt
twtfurl = http://example.org/twtxt.txt
check_following = True
use_pager = False
use_cache = True
porcelain = False
disclose_identity = False
character_limit = 140
character_warning = 140
limit_timeline = 20
timeline_update_interval = 10
timeout = 5.0
sorting = descending
pre_tweet_hook = "scp buckket@example.org:~/public_html/twtxt.txt {twtfilfile}"
post_tweet_hook = "scp {twtfilfile} buckket@example.org:~/public_html/twtxt.txt"
# post_tweet_hook = "tail -1 {twtfilfile} | cut -f2 | sed -e 's/^/twt=/' | curl -s -d @- -
↪d 'name=foo' -d 'password=bar' http://htwtxt.plomlompom.com/feeds"
# post_tweet_hook = "aws s3 cp {twtfilfile} s3://mybucket.org/twtxt.txt --acl public-
↪read --storage-class REDUCED_REDUNDANCY --cache-control 'max-age=60,public'"

[following]
bob = https://example.org/bob.txt
alice = https://example.org/alice.txt
```

## [twtxt]

| Option:                  | Type: | Default:   | Help:  |
|--------------------------|-------|------------|--|
| nick                     | TEXT  |            | your nick, will be displayed in your timeline  |
| twtfilfile               | PATH  |            | path to your local twtxt file                  |
| twtfurl                  | TEXT  |            | URL to your public twtxt file                  |
| check_following          | BOOL  | True       | try to resolve URLs when listing followings    |
| use_pager                | BOOL  | False      | use a pager (less) to display your timeline    |
| use_cache                | BOOL  | True       | cache remote twtxt files locally               |
| porcelain                | BOOL  | False      | style output in an easy-to-parse format        |
| disclose_identity        | BOOL  | False      | include nick and twtfurl in twtxt's user-agent |
| character_limit          | INT   | None       | shorten incoming tweets with more characters   |
| character_warning        | INT   | None       | warn when composed tweet has more characters   |
| limit_timeline           | INT   | 20         | limit amount of tweets shown in your timeline  |
| timeline_update_interval | INT   | 10         | time in seconds cache is considered up-to-date |
| timeout                  | FLOAT | 5.0        | maximal time a http request is allowed to take |
| sorting                  | TEXT  | descending | sort timeline either descending or ascending   |
| use_abs_time             | BOOL  | False      | use absolute datetimes in your timeline        |
| pre_tweet_hook           | TEXT  |            | command to be executed before tweeting         |
| post_tweet_hook          | TEXT  |            | command to be executed after tweeting          |

`pre_tweet_hook` and `post_tweet_hook` are very useful if you want to push your twtxt file to a remote (web) server. Check the example above tho see how it's used with `scp`.

## [followings]

This section holds all your followings as nick, URL pairs. You can edit this section manually or use the `follow/unfollow` commands of twtxt for greater comfort.

## twtxt file

The central component of sharing information, i.e. status updates, with twtxt is a simple text file containing all the status updates of a single user. This file is often referred as the *feed* of an user. The location of the twtxt file is configured in the twtxt section in the configuration file. See [Configuration](#).

## Format specification

The twtxt file contains one status per line, each of which is equipped with an RFC 3339 date-time string (with or without UTC offset) followed by a TAB character (`\t`) to separate it from the actual text. A specific ordering of the statuses is not mandatory.

The file must be encoded with UTF-8 and must use LF (`\n`) as line separators.

A status should consist of up to 140 characters, longer status updates are technically possible but discouraged. twtxt will warn the user if a newly composed status update exceeds this limit, and it will also shorten incoming status updates by default. Also note that a status may not contain any control characters.

Mentions are embedded within the text in either `@<source.nick source.url>` or `@<source.url>` format and should be expanded by the client, when rendering the tweets. The *source.url* is available to provide a way to discover new *twtxt.txt* files and distinguish between multiple users using the same nickname locally. The *source.url* can be interpreted as a TWTXT URI.

Take a look at this example file:

```
2016-02-04T13:30:00+01:00    You can really go crazy here! ()
2016-02-03T23:05:00+01:00    @<example http://example.org/twtxt.txt> welcome to twtxt!
2016-02-01T11:00:00+01:00    This is just another example.
2015-12-12T12:00:00+01:00    Fiat lux!
```

## Registry

Since twtxt is decentralized by design, features like the timeline are limited to the twtxt users followed by you. To provide a global search for mentions or hash tags, the client will be able to query so called registries.

A reference implementation is available at [twtxt-registry](#) (source) in nodejs and a demo instance is running at [twtxt-registry](#) (demo).

The client will support multiple registries at the same time, to circumvent possible single point of failures. The registries should sync each others user list by using the *users* endpoint.

## Format specification

### Writing to the registry

The responses return proper status code (200 OK) and the description of the status code or a detailed error message.

### Reading from the registry

The responses contain one status per line, each of which is equipped with nick of the poster followed by a TAB, the twtxturl of the poster followed by a TAB and an RFC 3339 date-time string followed by a TAB character (t) to separate it from the actual text.

```
NICK\tURL\tTIMESTAMP\tMESSAGE
```

### General rules

1. All lists are sorted by timestamp in descending order (most recent one first)
2. All lists support the *page* query parameter to get the next page of the result set.
3. The response must be encoded with UTF-8 and must use LF (\n) as line separators.
4. The columns are separated by a tab character (\t)

## API Endpoints

The url to the registry consists of its basepath (e.g. <https://registry.twtxt.org/api/>). The format (e.g. *plain*) is appended, because we might support json or xml format sooner or later.

The following parameters:

- basePath: <https://registry.twtxt.org/api/>
- format: *plain*
- endpoint: *tags/twtxt*

will call this url: <https://registry.twtxt.org/api/plain/tags/twtxt>.

## Add new User

Add a new Twtxt User to the Registry (Status Code is 200):

```
$ curl -X POST 'https://registry.twtxt.org/api/plain/users?url=https://example.org/
↳twtxt.txt&nickname=example'
OK
```

If it fails to add a new Twtxt User to the Registry (Status Code is 400):

```
$ curl -X POST 'https://registry.twtxt.org/api/plain/users?url=https://example.org/
↳twtxt.txt'
Bad Request: `nickname` is missing
```

## Latest tweets

See latest tweets in the Registry (e.g. [<https://registry.twtxt.org/api/plain/tweets>](https://registry.twtxt.org/api/plain/tweets)):

```
$ curl 'https://registry.twtxt.org/api/plain/tweets'
example https://example.org/twtxt.txt    2016-02-06T21:32:02.000Z    @erlehmnn is
↳messaging with timestamps in @bucket #twtxt :)
example https://example.org/twtxt.txt    2016-02-06T12:14:18.000Z    Simple nodejs
↳script to convert your twitter timeline to twtxt: https://t.co/txnWsC5jvA ( find my
↳#twtxt at https://t.co/uN1KDXwJ8B )
```

## Search for tweets

To query for tweets, which contain a specific word, use the tweets endpoint and the q query parameter.

```
$ curl 'https://registry.twtxt.org/api/plain/tweets?q=twtxt'
bucket https://bucket.org/twtxt.txt    2016-02-09T12:42:26.000Z    Do we need an
↳IRC channel for twtxt?
bucket https://bucket.org/twtxt.txt    2016-02-09T12:42:12.000Z    Good Morning,
↳twtxt-world!
```

## Query for mentions

To query for all tweets, which mention a specific user, use the mentions endpoint and the url query parameter.

```
$ curl 'https://registry.twtxt.org/api/plain/mentions?url=https://bucket.org/twtxt.
↳txt'
example https://example.org/twtxt.txt    2016-02-09T12:57:59.000Z    @<bucket
↳https://bucket.org/twtxt.txt> something like https://gitter.im/ or a freenode
↳channel?
example https://example.org/twtxt.txt    2016-02-08T22:51:47.000Z    @<bucket
↳https://bucket.org/twtxt.txt> looks nice ;)
```

## Query for tags

To query for all tweets, which contain a specific tag like *#twtxt*, use the tags endpoint and prepend the tag.

```
$ curl 'https://registry.twtxt.org/api/plain/tags/twtxt'
example https://example.org/twtxt.txt    2016-02-06T21:32:02.000Z    @erlehmnn is
↳messaging with timestamps in @bucket #twtxt :)
example https://example.org/twtxt.txt    2016-02-06T12:14:18.000Z    Simple nodejs
↳script to convert your twitter timeline to twtxt: https://t.co/txnWsC5jvA ( find my
↳#twtxt at https://t.co/uN1KDXwJ8B )
```

## Query for users

To query for a user list, use the users endpoint and refine with the q query parameter.

```
$ curl 'https://registry.twtxt.org/api/plain/users?q=example'
example https://example.org/twtxt.txt    2016-02-09T12:42:26.000Z
example42 https://example.org/42.twtxt.txt    2016-02-10T13:20:10.000Z
```

## Discoverability

Because of the decentral nature of twtxt it can be hard to find new peers to follow, or even know who is following one's own feed. The later being a problem because right now mentions only show up when you actively follow the feed the mention originated from.

To solve this issue, besides the usage of *registries*, twtxt is using a specially crafted User-Agent string, when making outgoing HTTP requests. This then allows other users to search their webserver's log file for those strings and find out who is consuming their content.

---

**Note:** Implementing a so called linkback mechanism to actively notify someone explicitly about incoming mentions is currently [being discussed on GitHub](#).

---

The format twtxt is using is as follows:

```
twtxt/<version> (+<source.url>; @<source.nick>)
```

For example:

```
twtxt/1.2.3 (+https://example.com/twtxt.txt; @somebody)
```

Other clients are encouraged to use the same format.





## CHAPTER 2

---

### Community

---

- twtxt IRC channel: **#twtxt** on [irc.freenode.net](https://irc.freenode.net)



This part of the documentation describes the modules, classes, functions and other source code specific details of twtxt.

## API

This chapter documents twtxts API and source code internals.

### Models

**class** `models.Tweet` (*text*, *created\_at=None*, *source=None*)

A `Tweet` represents a single tweet.

#### Parameters

- **text** (*str*) – text of the tweet in raw format
- **created\_at** (*datetime*) – (optional) when the tweet was created, defaults to `now()` when no value is given
- **source** (*Source*) – (optional) the `Source` the tweet is from

#### **absolute\_datetime**

Return human-readable absolute time string.

#### **relative\_datetime**

Return human-readable relative time string.

**class** `models.Source` (*nick*, *url=None*, *file=None*)

A `Source` represents a twtxt feed, remote as well as local.

#### Parameters

- **nick** (*str*) – nickname of twtxt user
- **url** (*str*) – URL to remote twtxt file

- **file** (*str*) – path to local twtxt file

## Config

Every supported option of twtxt is made available as a property of the `Config` object. To find out the meaning of those check [Configuration](#).

**class** `config.Config` (*config\_file*, *cfg*)  
`Config` interacts with the configuration file.

### Parameters

- **config\_file** (*str*) – full path to the loaded config file
- **cfg** (*ConfigParser*) – a `ConfigParser` object with config loaded

**add\_source** (*source*)  
Adds a new `Source` to the config's following section.

**build\_default\_map** ()  
Maps config options to the default values used by click, returns `dict`.

**check\_config\_sanity** ()  
Checks if the given values in the config file are sane.

**classmethod create\_config** (*cfgfile*, *nick*, *twtf*, *twurl*, *disclose\_identity*, *add\_news*)  
Create a new config file at the default location.

### Parameters

- **cfgfile** (*str*) – path to the config file
- **nick** (*str*) – nickname to use for own tweets
- **twtf** (*str*) – path to the local twtxt file
- **twurl** (*str*) – URL to the remote twtxt file
- **disclose\_identity** (*bool*) – if true the users id will be disclosed
- **add\_news** (*bool*) – if true follow twtxt news feed

**classmethod discover** ()  
Make a guess about the config file location an try loading it.

**following**  
A `list` of all `Source` objects.

**classmethod from\_file** (*file*)  
Try loading given config file.

**Parameters** **file** (*str*) – full path to the config file to load

**get\_source\_by\_nick** (*nick*)  
Returns the `Source` of the given nick.

**Parameters** **nick** (*str*) – nickname for which will be searched in the config

**options**  
A `dict` of all config options.

**remove\_source\_by\_nick** (*nick*)  
Removes a `Source` form the config's following section.

**Parameters** **nick** (*str*) – nickname for which will be searched in the config

**write\_config()**

Writes *self.cfg* to *self.config\_file*.



**t**

twtxt, [15](#)





### A

`absolute_datetime` (`models.Tweet` attribute), [15](#)  
`add_source()` (`config.Config` method), [16](#)

### B

`build_default_map()` (`config.Config` method), [16](#)

### C

`check_config_sanity()` (`config.Config` method), [16](#)  
`Config` (class in `config`), [16](#)  
`create_config()` (`config.Config` class method), [16](#)

### D

`discover()` (`config.Config` class method), [16](#)

### F

`following` (`config.Config` attribute), [16](#)  
`from_file()` (`config.Config` class method), [16](#)

### G

`get_source_by_nick()` (`config.Config` method), [16](#)

### O

`options` (`config.Config` attribute), [16](#)

### R

`relative_datetime` (`models.Tweet` attribute), [15](#)  
`remove_source_by_nick()` (`config.Config` method), [16](#)

### S

`Source` (class in `models`), [15](#)

### T

`Tweet` (class in `models`), [15](#)  
`twtxt` (module), [15](#)

### W

`write_config()` (`config.Config` method), [16](#)