twtxt Documentation

Release 1.2.1

buckket

February 16, 2016

Contents

Release: v1.2.1.

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.

CHAPTER 1

User Guide

1.1 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.

1.1.1 Demonstration

1.1.2 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`!

1.2 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

1.2.1 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!

1.2.2 Development version

Clone the git repository:

```
$ git clone https://github.com/buckket/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/
```

1.3 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. Have a look at the README.rst 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 twtxt
    file [~/twtxt.txt]: <TWTXTX FILE LOCATION>

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

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

The quickstart wizard prompts for the following configuration values:

- Your desired nick name (*<NICKNAME>*)
- The desired location for your twtxt file (*<TWTXTX FILE LOCATION>*)
- If you want to follow the twtxt news feed

The configurations can easily be changed in the twtxt configuration file. See *Configuration*.

1.4 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:

1.4.1 Follow a source

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

1.4.2 List all sources you're following

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

1.4.3 Unfollow a source

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

1.4.4 Post a status update

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

1.4.5 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?
```

1.4.6 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!
```

1.5 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. OSX 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
twtfile = ~/twtxt.txt
twturl = http://example.org/twtxt.txt
check_following = True
use_pager = False
use_cache = True
porcelain = False
disclose_identity = False
character_limit = 140
limit_timeline = 20
timeout = 5.0
sorting = descending
pre_tweet_hook = "scp buckket@example.org:~/public_html/twtxt.txt {twtfile}"
post_tweet_hook = "scp {twtfile} buckket@example.org:~/public_html/twtxt.txt"
# post_tweet_hook = "tail -1 {twtfile} | cut -f2 | sed -e 's/^/twt=/'| curl -s -d @- -d 'name=foo' -
# post_tweet_hook = "aws s3 cp {twtfile} s3://mybucket.org/twtxt.txt --acl public-read --storage-clas

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

1.5.1 [twtxt]

Option:	Type:	Default:	Help:
nick	TEXT		your nick, will be displayed in your timeline
twtfile	PATH		path to your local twtxt file
twturl	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 twturl in twtxt's user-agent
character_limit	INT	None	limit amount of characters a tweet can have
limit_timeline	INT	20	limit amount of tweets shown in your timeline
timeout	FLOAT	5.0	maximal time a http request is allowed to take
sorting	TEXT	descending	sort timeline either descending or ascending
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`.

1.5.2 [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.

1.6 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*.

1.6.1 Format specification

The twtxt file contains one status per line, each of which is equipped with an ISO 8601 date/time string 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+01 You can really go crazy here! ()
2016-02-03T23:05+01 @<example http://example.org/twtxt.txt> welcome to twtxt!
2016-02-01T11:00+01 This is just another example.
2015-12-12T12:00+01 Fiat lux!
```

1.7 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.

1.7.1 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 reference to the poster followed by a TAB and an ISO 8601 date/time string followed by a TAB character (\t) to separate it from the actual text.

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.

1.7.2 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*.

1.7.3 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=
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
```

See latest tweets in the Registry (e.g. <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        @erlehmann is messing
@<example https://example.org/twtxt.txt>        2016-02-06T12:14:18.000Z        Simple nodejs script
```

1.7.4 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
@<buckket https://buckket.org/twtxt.txt>        2016-02-09T12:42:26.000Z        Do we need an IRC cha
@<buckket https://buckket.org/twtxt.txt>        2016-02-09T12:42:12.000Z        Good Morning, twtxt-w
```

1.7.5 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://buckket.org/twtxt.txt
@<example https://example.org/twtxt.txt>        2016-02-09T12:57:59.000Z        @<buckket https://buc
@<example https://example.org/twtxt.txt>        2016-02-08T22:51:47.000Z        @<buckket https://buc
```

1.7.6 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        @erlehmann is messing
@<example https://example.org/twtxt.txt>        2016-02-06T12:14:18.000Z        Simple nodejs script
```

1.7.7 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        example
<@example https://example.org/42.twtxt.txt>     2016-02-10T13:20:10.000Z        example42
```

CHAPTER 2

Community

- twtxt IRC channel: **#twtxt** on irc.freenode.net

CHAPTER 3

API Reference

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

3.1 API

This chapter documents twtxts API and source code internals.

3.1.1 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

`relative_datetime`
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

3.1.2 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.

classmethod create_config(*nick*, *twtfile*, *add_news*)
Create a new config file at the default location.

Parameters

- **nick** (*str*) – nickname to use for own tweets
- **twtfile** (*str*) – path to the local twtxt file
- **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

A

B

C

D

F

G

O

R

S

T

W