
py8chan Documentation

Release 0.0.3

Antonizoon Overtwater

Nov 26, 2018

Contents

1	General Documentation	3
1.1	Tutorial	3
1.2	Changes from the original py4chan	4
2	API Documentation	7
2.1	py8chan – 8chan Python Library	7
2.2	py8chan.Board – 8chan Boards	8
2.3	py8chan.Thread – 8chan Threads	10
2.4	py8chan.Post – 8chan Post	11
2.5	py8chan.File – 8chan File	14
	Python Module Index	17

py8chan is a Python library that gives access to the 8chan API and an object-oriented way to browse and get board and thread information quickly and easily.

py8chan is based on [BASC-py4chan](#), a 4chan API wrapper that was adopted and extended by the [Bibliotheca Anonoma](#).

The py8chan repository is located [on Github](#), where pull requests and issues can be submitted.

Getting Help If you want help, or you have some trouble using this library, our primary IRC channel is [#bibanon on irc.rizon.net](#). Simply head in there and talk to dan or antonizoon. Otherwise, you can put a issue on our [Github Issue Tracker](#) and we'll respond as soon as we can!

1.1 Tutorial

When using `py8chan`, it can be a bit hard to find where to begin. Here, we run through how to create and use the various objects available in this module.

1.1.1 Boards

`py8chan.Board` is the first thing you create when using `py8chan`. Everything else is created through that class. The most basic way to create a board is as below:

```
board = py8chan.Board('v')
```

This creates a `py8chan.Board` object that you can then use to create `py8chan.Thread` and `py8chan.Post` objects.

But what sort of things does a `py8chan.Board` object let you do?

Here's a short code snippet of us printing out how many threads are active on a board:

```
board = py8chan.Board('v')
thread_ids = board.get_all_thread_ids()
str_thread_ids = [str(id) for id in thread_ids] # need to do this so str.join below_
↳works
print('There are', len(all_ids), 'active threads on /tg/:', ', '.join(str_thread_ids))
```

1.1.2 Threads

Listing how many threads exist on a board is all well and good, but most people want to actually get threads and do things with them. Here, we'll describe how to do that.

All `py8chan.Thread` objects are created by a `py8chan.Board` object, using one of the `py8chan.Board.get_thread()` methods.

For this example, we have a user ask us about “thread 1234”, and we return information about it:

```
thread_id = 1234
board = py8chan.Board('v')

if board.thread_exists(thread_id):
    thread = board.get_thread(thread_id)

    # print thread information
    print('Thread', thread_id)
    if thread.closed:
        print(' is closed')
    if thread.sticky:
        print(' is a sticky')

    # information from the OP
    topic = thread.topic
    print(' is named:', topic.subject)
    print(' and was made by:', name, email)
```

1.2 Changes from the original py4chan

Since Edgeworth has gone MIA, [BASC](#) has adopted the project and made the following improvements.

1.2.1 Changes by antonizoon

- **4chan Link Structure Update** - 4chan has heavily reformed it’s link structure, finally removing the strange folder structure inherited from the Futaba Channel.
- **4chan cdn Link update** - To save money on bandwidth. 4chan has changed it’s image/thumbnaill/json/css servers to a domain name with fewer characters.
- **Thread Class:** new `filenames()` function that return the filenames of all files (not thumbnails) in a thread.
- **Thread Class:** new `thumbnames()` function that return the filenames of all thumbnails in a thread.
 - **Post Class:** new `image_fname` and `thumbnail_fname` properties, designed for Thread Class `filenames()` and `thumbnames()`.
- **Actual API Documentation** - Real documentation on using the py-4chan library is a must. For some people, it is rocket science.

1.2.2 Changes by Anorov

- **Anorov’s underscore_function_notation** - Even I have to say that CamelCase is beginning to suck, so we’ve adopted Anorov’s function notation for py4chan. This breaks API compatibility with the original py-4chan, but just use `find/replace` to change your functions.
- **Break up classes into separate files.** - Makes the code much cleaner.
- Thread Class: `expand()` function, used to display omitted posts and images. Used by `all_posts()`.
- Thread Class: `semantic_thread_url()` function, used to obtain 4chan’s new URL format, which tacks on the thread title (obtained from `slug()`).

- Post Class: `comment()` has been modified to use `clean_comment_body()` when returning a comment. The raw text from the 4chan API can still be obtained from `orig_comment()`.
 - Util Class: `clean_comment_body()` function, which converts all HTML tags and entities within 4chan comments into human-readable text equivalents.(e.g. `
` to a newline, `<a href>` into a raw link)
- Board Class: `_get_json()` function, which dumps the raw JSON from the 4chan API.
- A whole host of new Catalog parsing functions:
 - Board Class: `refresh_cache()` and `clear_cache()` - Get the latest Catalog of all threads in the board, or clear the current cache.
 - Board Class: `get_threads(page)` - Get a list of all threads on a certain page. (Pages are now indexed starting from 1).
 - Board Class: `get_all_thread_ids()` - Get a list of all thread IDs on the board.
 - Board Class: `get_all_threads()` - Return all threads on all pages in the board.

1.2.3 Changes by Daniel Oaks

- **ReadTheDocs Documentation** - Splitting the documentation out to [ReadTheDocs](#), using [Sphinx](#) to generate nice, useful docs!

2.1 py8chan – 8chan Python Library

py8chan gives access to 8chan from a clean Python interface.

2.1.1 Basic Usage

4chan Python Library.

BASC-py4chan is a Python library that gives access to the 4chan API and an object-oriented way to browse and get board and thread information quickly and easily.

2.1.2 Methods

`py8chan.get_boards` (*board_name_list*, *args, **kwargs)

Given a list of boards, return `basc_py4chan.Board` objects.

Parameters `board_name_list` (*list*) – List of board names to get, eg: ['b', 'tg']

Returns Requested boards.

Return type dict of `basc_py4chan.Board`

`py8chan.get_all_boards` (*args, **kwargs)

Returns every board on 4chan.

Returns All boards.

Return type dict of `basc_py4chan.Board`

2.2 py8chan.Board – 8chan Boards

`py8chan.Board` provides access to a 8chan board including checking if threads exist, retrieving appropriate `py8chan.Thread` objects, and returning lists of all the threads that exist on the given board.

2.2.1 Example

Here is a sample application that grabs and uses Board information:

```
from __future__ import print_function
import py8chan

board = py8chan.Board('tg')
thread_ids = board.get_all_thread_ids()
str_thread_ids = [str(id) for id in thread_ids] # need to do this so str.join below
↪works
print('There are', len(all_ids), 'active threads on /tg/:', ', '.join(str_thread_ids))
```

2.2.2 Basic Usage

class `py8chan.Board` (*board_name*, *https=False*, *session=None*)

Represents a 4chan board.

name

Name of this board, such as `tg` or `k`.

Type `str`

name

Name of the board, such as “`tg`” or “`etc`”.

Type `string`

title

Board title, such as “Animu and Mango”.

Type `string`

is_worksafe

Whether this board is worksafe.

Type `bool`

page_count

How many pages this board has.

Type `int`

threads_per_page

How many threads there are on each page.

Type `int`

2.2.3 Methods

`Board.__init__` (*board_name*, *https=False*, *session=None*)

Creates a `basc_py4chan.Board` object.

Parameters

- **board_name** (*string*) – Name of the board, such as “tg” or “etc”.
- **https** (*bool*) – Whether to use a secure connection to 4chan.
- **session** – Existing requests.session object to use instead of our current one.

`Board.thread_exists(thread_id)`

Check if a thread exists or has 404'd.

Parameters `thread_id` (*int*) – Thread ID

Returns Whether the given thread exists on this board.

Return type `bool`

`Board.get_thread(thread_id, update_if_cached=True, raise_404=False)`

Get a thread from 4chan via 4chan API.

Parameters

- **thread_id** (*int*) – Thread ID
- **update_if_cached** (*bool*) – Whether the thread should be updated if it's already in our cache
- **raise_404** (*bool*) – Raise an Exception if thread has 404'd

Returns Thread object

Return type `basc_py4chan.Thread`

`Board.get_threads(page=0)`

Returns all threads on a certain page.

Gets a list of Thread objects for every thread on the given page. If a thread is already in our cache, the cached version is returned and `thread.want_update` is set to True on the specific thread object.

Pages on 8chan/vichan are indexed from 0 onwards. (not 1 as in modern 4chan: 4chan used to start from 0)

Parameters `page` (*int*) – Page to request threads for. Defaults to the first page.

Returns List of Thread objects representing the threads on the given page.

Return type list of `basc_py4chan.Thread`

`Board.get_all_threads(expand=False)`

Return every thread on this board.

If not expanded, result is same as `get_threads` run across all board pages, with last 3-5 replies included.

Uses the catalog when not expanding, and uses the flat thread ID listing at `/{board}/threads.json` when expanding for more efficient resource usage.

If expanded, all data of all threads is returned with no omitted posts.

Parameters `expand` (*bool*) – Whether to download every single post of every thread. If enabled, this option can be very slow and bandwidth-intensive.

Returns List of Thread objects representing every thread on this board.

Return type list of `basc_py4chan.Thread`

`Board.get_all_thread_ids()`

Return the ID of every thread on this board.

Returns List of IDs of every thread on this board.

Return type list of ints

`Board.refresh_cache (if_want_update=False)`
Update all threads currently stored in our cache.

`Board.clear_cache ()`
Remove everything currently stored in our cache.

2.3 py8chan.Thread – 8chan Threads

`py8chan.Thread` allows for standard access to a 8chan thread, including listing all the posts in the thread, information such as whether the thread is locked and stickied, and lists of attached file URLs or thumbnails.

2.3.1 Basic Usage

class `py8chan.Thread (board, id)`

Represents a thread.

closed

Whether the thread has been closed.

Type bool

sticky

Whether this thread is a ‘sticky’.

Type bool

topic

Topic post of the thread, the OP.

Type `py8chan.Post`

posts

List of all posts in the thread, including the OP.

Type list of `py8chan.Post`

all_posts

List of all posts in the thread, including the OP and any omitted posts.

Type list of `py8chan.Post`

url

URL of the thread, not including semantic slug.

Type string

Undefined Attributes

Type Not implemented in 8chan API. Do not use.

replies and images

Infuriatingly, the OP post in a thread

doesn't list how many replies there are in a thread.

semantic_url

URL of this post, with the thread's ‘semantic’ component.

Type string

semantic_slug

This post's 'semantic slug'.

Type string

2.3.2 Methods

Thread objects are not instantiated directly, but instead through the appropriate `py8chan.Board` methods such as `py8chan.Board.get_thread()`.

`Thread.files()`

Returns the URLs of all files attached to posts in the thread.

`Thread.thumbs()`

Returns the URLs of all thumbnails in the thread.

`Thread filenames()`

Returns the filenames of all files attached to posts in the thread.

`Thread.thumbnames()`

Returns the filenames of all thumbnails in the thread.

`Thread.update(force=False)`

Fetch new posts from the server.

Parameters `force` (*bool*) – Force a thread update, even if thread has 404'd.

Returns How many new posts have been fetched.

Return type int

`Thread.expand()`

If there are omitted posts, update to include all posts.

2.4 py8chan.Post – 8chan Post

`py8chan.Post` allows for standard access to a 8chan post.

2.4.1 Example

Here is a sample application that grabs and prints `py8chan.Thread` and `py8chan.Post` information:

```
# credits to Anarov for improved example
from __future__ import print_function
import py8chan

# get the board we want
board = py8chan.Board('v')

# select the first thread on the board
all_thread_ids = board.get_all_thread_ids()
first_thread_id = all_thread_ids[0]
thread = board.get_thread(first_thread_id)
```

(continues on next page)

(continued from previous page)

```
# print thread information
print(thread)
print('Sticky?', thread.sticky)
print('Closed?', thread.closed)
print('Replies:', len(thread.replies))

# print topic post information
topic = thread.topic
print('Topic Repr', topic)
print('Postnumber', topic.post_number)
print('Timestamp', topic.timestamp)
print('Datetime', repr(topic.datetime))
print('Filemd5hex', topic.file_md5_hex)
print('Fileurl', topic.file_url)
print('Subject', topic.subject)
print('Comment', topic.comment)
print('Thumbnailurl', topic.thumbnail_url)
```

2.4.2 Basic Usage

class `py8chan.Post` (*thread, data*)

Represents a 4chan post.

post_id

ID of this post. Eg: 123123123, 456456456.

Type `int`

poster_id

Poster ID.

Type `int`

name

Poster's name.

Type `string`

email

Poster's email.

Type `string`

tripcode

Poster's tripcode.

Type `string`

subject

Subject of this post.

Type `string`

comment

This comment, with the `<wbr>` tag removed.

Type `string`

html_comment

Original, direct HTML of this comment.

Type string

text_comment

Plaintext version of this comment.

Type string

is_op

Whether this is the OP (first post of the thread)

Type bool

timestamp

Unix timestamp for this post.

Type int

datetime

Datetime time of this post.

Type `datetime.datetime`

first_file

The first file of the post.

Type *File*

all_files

Returns the File objects of all files in the post.

Type *File*

extra_files

Returns the File objects of all extra files in the post, if any.

Type *File*

has_file

Whether this post has a file attached to it.

Type bool

has_extra_files

Whether this post has more than one file attached to it.

Type bool

url

URL of this post.

Type string

Undefined Attributes (Not implemented in 8chan API. Do not use.): `poster_id` (int): Poster ID. `file_deleted` (bool): Whether the file attached to this post was deleted after being posted. `semantic_url` (string): URL of this post, with the thread's 'semantic' component. `semantic_slug` (string): This post's 'semantic slug'.

Post objects are not instantiated directly, but through a `py8chan.Thread` object with an attribute like `py8chan.Thread.all_posts`.

2.5 py8chan.File – 8chan File

`py8chan.Post` allows for standard access to a 4chan file. This provides programs with a complete File object that contains all metadata about the 4chan file, and makes migration easy if 4chan ever makes multiple files in one Post possible (as 8chan does).

2.5.1 Basic Usage

class `py8chan.File` (*post*, *data*)

Represents File objects and their thumbnails.

Constructor: `post` (`py8chan.Post`) - parent Post object. `data` (dict) - The post or `extra_files` dict from the 8chan API.

file_md5

MD5 hash of the file attached to this post.

Type string

file_md5_hex

Hex-encoded MD5 hash of the file attached to this post.

Type string

filename_original

Original name of the file attached to this post.

Type string

filename

Filename of the file attached to this post.

Type string

file_url

URL of the file attached to this post.

Type string

file_extension

Extension of the file attached to this post. Eg: `png`, `webm`, etc.

Type string

file_size

Size of the file attached to this post.

Type int

file_width

Width of the file attached to this post.

Type int

file_height

Height of the file attached to this post.

Type int

thumbnail_width

Width of the thumbnail attached to this post.

Type int

thumbnail_height

Height of the thumbnail attached to this post.

Type int

thumbnail_fname

Filename of the thumbnail attached to this post.

Type string

thumbnail_url

URL of the thumbnail attached to this post.

Type string

File objects are not instantiated directly, but through a *py8chan.File* object with an attribute like *py8chan.Post.first_file*.

p

py8chan, 7

Symbols

`__init__()` (*py8chan.Board* method), 8

A

`all_files` (*py8chan.Post* attribute), 13
`all_posts` (*py8chan.Thread* attribute), 10

B

`Board` (*class in py8chan*), 8

C

`clear_cache()` (*py8chan.Board* method), 10
`closed` (*py8chan.Thread* attribute), 10
`comment` (*py8chan.Post* attribute), 12

D

`datetime` (*py8chan.Post* attribute), 13

E

`email` (*py8chan.Post* attribute), 12
`expand()` (*py8chan.Thread* method), 11
`extra_files` (*py8chan.Post* attribute), 13

F

`File` (*class in py8chan*), 14
`file_extension` (*py8chan.File* attribute), 14
`file_height` (*py8chan.File* attribute), 14
`file_md5` (*py8chan.File* attribute), 14
`file_md5_hex` (*py8chan.File* attribute), 14
`file_size` (*py8chan.File* attribute), 14
`file_url` (*py8chan.File* attribute), 14
`file_width` (*py8chan.File* attribute), 14
`filename` (*py8chan.File* attribute), 14
`filename_original` (*py8chan.File* attribute), 14
`filenames()` (*py8chan.Thread* method), 11
`files()` (*py8chan.Thread* method), 11
`first_file` (*py8chan.Post* attribute), 13

G

`get_all_boards()` (*in module py8chan*), 7
`get_all_thread_ids()` (*py8chan.Board* method), 9
`get_all_threads()` (*py8chan.Board* method), 9
`get_boards()` (*in module py8chan*), 7
`get_thread()` (*py8chan.Board* method), 9
`get_threads()` (*py8chan.Board* method), 9

H

`has_extra_files` (*py8chan.Post* attribute), 13
`has_file` (*py8chan.Post* attribute), 13
`html_comment` (*py8chan.Post* attribute), 12

I

`is_op` (*py8chan.Post* attribute), 13
`is_worksafe` (*py8chan.Board* attribute), 8

N

`name` (*py8chan.Board* attribute), 8
`name` (*py8chan.Post* attribute), 12

P

`page_count` (*py8chan.Board* attribute), 8
`Post` (*class in py8chan*), 12
`post_id` (*py8chan.Post* attribute), 12
`poster_id` (*py8chan.Post* attribute), 12
`posts` (*py8chan.Thread* attribute), 10
`py8chan` (*module*), 7

R

`refresh_cache()` (*py8chan.Board* method), 10

S

`semantic_slug` (*py8chan.Thread* attribute), 11
`semantic_url` (*py8chan.Thread* attribute), 10
`sticky` (*py8chan.Thread* attribute), 10
`subject` (*py8chan.Post* attribute), 12

T

`text_comment` (*py8chan.Post attribute*), 13
`Thread` (*class in py8chan*), 10
`thread_exists()` (*py8chan.Board method*), 9
`threads_per_page` (*py8chan.Board attribute*), 8
`thumbnail_fname` (*py8chan.File attribute*), 15
`thumbnail_height` (*py8chan.File attribute*), 14
`thumbnail_url` (*py8chan.File attribute*), 15
`thumbnail_width` (*py8chan.File attribute*), 14
`thumbnames()` (*py8chan.Thread method*), 11
`thumbs()` (*py8chan.Thread method*), 11
`timestamp` (*py8chan.Post attribute*), 13
`title` (*py8chan.Board attribute*), 8
`topic` (*py8chan.Thread attribute*), 10
`tripcode` (*py8chan.Post attribute*), 12

U

`update()` (*py8chan.Thread method*), 11
`url` (*py8chan.Post attribute*), 13
`url` (*py8chan.Thread attribute*), 10