
BASC-py4chan Documentation

Release 0.6.3

Antonizoon Overtwater

Nov 25, 2018

Contents

1	General Documentation	3
1.1	Tutorial	3
1.2	Changes from the original py4chan	4
2	API Documentation	7
2.1	basc_py4chan – 4chan Python Library	7
2.2	basc_py4chan.Board – 4chan Boards	8
2.3	basc_py4chan.Thread – 4chan Threads	10
2.4	basc_py4chan.Post – 4chan Post	11
2.5	basc_py4chan.File – 4chan File	14
	Python Module Index	17

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.

Originally written by [Edgeworth](#), the library has been adopted and extended by [Bibliotheca Anonoma](#).

Warning: If you have an old application written to use the original py4chan, [Bibliotheca Anonoma](#) also maintains a [py-4chan fork](#) on legacy support, only to be updated for URL changes without any new features. This fork is also linked to the [original PyPi package](#), and updating py-4chan using pip will give you the latest version of this fork.

However, we recommend that **all users** switch to the new BASC-py4chan. This module is more Pythonic, has better support, documentation, and will be gaining new features.

The BASC-py4chan 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 BASC-py4chan, 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

basc_py4chan.Board is the first thing you create when using BASC-py4chan. Everything else is created through that class. The most basic way to create a board is as below:

```
board = basc_py4chan.Board('tg')
```

This creates a *basc_py4chan.Board* object that you can then use to create *basc_py4chan.Thread* and *basc_py4chan.Post* objects.

But what sort of things does a *basc_py4chan.Board* object let you do?

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

```
board = basc_py4chan.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))
```

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 *basc_py4chan.Thread* objects are created by a *basc_py4chan.Board* object, using one of the *basc_py4chan.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 = basc_py4chan.Board('tg')

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, [The Bibliotheca Anonoma](#) 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 `basc_py4chan` – 4chan Python Library

`basc_py4chan` gives access to 4chan 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

`basc_py4chan.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`

`basc_py4chan.get_all_boards(*args, **kwargs)`

Returns every board on 4chan.

Returns All boards.

Return type dict of `basc_py4chan.Board`

2.2 `basc_py4chan.Board` – 4chan Boards

`basc_py4chan.Board` provides access to a 4chan board including checking if threads exist, retrieving appropriate `basc_py4chan.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 basc_py4chan

board = basc_py4chan.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 `basc_py4chan.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=1)`

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 4chan are indexed from 1 onwards.

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 `basc_py4chan.Thread` – 4chan Threads

`basc_py4chan.Thread` allows for standard access to a 4chan 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 `basc_py4chan.Thread (board, id)`

Represents a 4chan thread.

closed

Whether the thread has been closed.

Type bool

sticky

Whether this thread is a ‘sticky’.

Type bool

archived

Whether the thread has been archived.

Type bool

bumplimit

Whether the thread has hit the bump limit.

Type bool

imagelimit

Whether the thread has hit the image limit.

Type bool

custom_spoiler

Number of custom spoilers in the thread (if the board supports it)

Type int

topic

Topic post of the thread, the OP.

Type `basc_py4chan.Post`

posts

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

Type list of `basc_py4chan.Post`

all_posts

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

Type list of *basc_py4chan.Post*

url

URL of the thread, not including semantic slug.

Type string

semantic_url

URL of the thread, with the semantic slug.

Type string

semantic_slug

The ‘pretty URL slug’ assigned to this thread by 4chan.

Type string

2.3.2 Methods

Thread objects are not instantiated directly, but instead through the appropriate *basc_py4chan.Board* methods such as *basc_py4chan.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 basc_py4chan.Post – 4chan Post

basc_py4chan.Post allows for standard access to a 4chan post.

2.4.1 Example

Here is a sample application that grabs and prints *basc_py4chan.Thread* and *basc_py4chan.Post* information:

```
# credits to Anarov for improved example
from __future__ import print_function
import basc_py4chan

# get the board we want
board = basc_py4chan.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)

# 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('Subject', topic.subject)
print('Comment', topic.comment)

# file information
for f in first_thread.file_objects():
    print('Filename', f.filename)
    print('  Filemd5hex', f.file_md5_hex)
    print('  Fileurl', f.file_url)
    print('  Thumbnailurl', f.thumbnail_url)
    print()
```

2.4.2 Basic Usage

class basc_py4chan.**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

spoiler
Whether the attached file is spoiled.
Type bool

timestamp
Unix timestamp for this post.
Type int

datetime
Datetime time of this post.
Type datetime.datetime

first_file
The File object associated with this post.
Type py8chan.File

has_file
Whether this post has a file attached to it.
Type bool

url
URL of this post.
Type string

semantic_url
URL of this post, with the thread's 'semantic' component.
Type string

semantic_slug
This post's 'semantic slug'.

Type string

Post objects are not instantiated directly, but through a `basc_py4chan.Thread` object with an attribute like `basc_py4chan.Thread.all_posts`.

2.5 basc_py4chan.File – 4chan File

`basc_py4chan.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 `basc_py4chan.File` (*post*, *data*)

Represents File objects and their thumbnails. Constructor:

`post` (`py4chan.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

Name of the file attached to this post.

Type string

filename_original

Original name 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

file_deleted

Whether the file attached to this post was deleted after being posted.

Type bool

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 *basc_py4chan.File* object with an attribute like *basc_py4chan.Post.first_file*.

b

`base_py4chan`, 7

Symbols

`__init__()` (basc_py4chan.Board method), 8

A

`all_posts` (basc_py4chan.Thread attribute), 10

`archived` (basc_py4chan.Thread attribute), 10

B

`basc_py4chan` (module), 7

`Board` (class in basc_py4chan), 8

`bumplimit` (basc_py4chan.Thread attribute), 10

C

`clear_cache()` (basc_py4chan.Board method), 10

`closed` (basc_py4chan.Thread attribute), 10

`comment` (basc_py4chan.Post attribute), 13

`custom_spoiler` (basc_py4chan.Thread attribute), 10

D

`datetime` (basc_py4chan.Post attribute), 13

E

`email` (basc_py4chan.Post attribute), 12

`expand()` (basc_py4chan.Thread method), 11

F

`File` (class in basc_py4chan), 14

`file_deleted` (basc_py4chan.File attribute), 15

`file_extension` (basc_py4chan.File attribute), 14

`file_height` (basc_py4chan.File attribute), 14

`file_md5` (basc_py4chan.File attribute), 14

`file_md5_hex` (basc_py4chan.File attribute), 14

`file_size` (basc_py4chan.File attribute), 14

`file_url` (basc_py4chan.File attribute), 14

`file_width` (basc_py4chan.File attribute), 14

`filename` (basc_py4chan.File attribute), 14

`filename_original` (basc_py4chan.File attribute), 14

`filenames()` (basc_py4chan.Thread method), 11

`files()` (basc_py4chan.Thread method), 11

`first_file` (basc_py4chan.Post attribute), 13

G

`get_all_boards()` (in module basc_py4chan), 7

`get_all_thread_ids()` (basc_py4chan.Board method), 9

`get_all_threads()` (basc_py4chan.Board method), 9

`get_boards()` (in module basc_py4chan), 7

`get_thread()` (basc_py4chan.Board method), 9

`get_threads()` (basc_py4chan.Board method), 9

H

`has_file` (basc_py4chan.Post attribute), 13

`html_comment` (basc_py4chan.Post attribute), 13

I

`imagelimit` (basc_py4chan.Thread attribute), 10

`is_op` (basc_py4chan.Post attribute), 13

`is_worksafe` (basc_py4chan.Board attribute), 8

N

`name` (basc_py4chan.Board attribute), 8

`name` (basc_py4chan.Post attribute), 12

P

`page_count` (basc_py4chan.Board attribute), 8

`Post` (class in basc_py4chan), 12

`post_id` (basc_py4chan.Post attribute), 12

`poster_id` (basc_py4chan.Post attribute), 12

`posts` (basc_py4chan.Thread attribute), 10

R

`refresh_cache()` (basc_py4chan.Board method), 10

S

`semantic_slug` (basc_py4chan.Post attribute), 13

`semantic_slug` (basc_py4chan.Thread attribute), 11

`semantic_url` (basc_py4chan.Post attribute), 13

`semantic_url` (basc_py4chan.Thread attribute), 11

`spoiler` (basc_py4chan.Post attribute), 13

sticky (basc_py4chan.Thread attribute), [10](#)

subject (basc_py4chan.Post attribute), [13](#)

T

text_comment (basc_py4chan.Post attribute), [13](#)

Thread (class in basc_py4chan), [10](#)

thread_exists() (basc_py4chan.Board method), [9](#)

threads_per_page (basc_py4chan.Board attribute), [8](#)

thumbnail_fname (basc_py4chan.File attribute), [15](#)

thumbnail_height (basc_py4chan.File attribute), [15](#)

thumbnail_url (basc_py4chan.File attribute), [15](#)

thumbnail_width (basc_py4chan.File attribute), [15](#)

thumbnames() (basc_py4chan.Thread method), [11](#)

thumbs() (basc_py4chan.Thread method), [11](#)

timestamp (basc_py4chan.Post attribute), [13](#)

title (basc_py4chan.Board attribute), [8](#)

topic (basc_py4chan.Thread attribute), [10](#)

tripcode (basc_py4chan.Post attribute), [12](#)

U

update() (basc_py4chan.Thread method), [11](#)

url (basc_py4chan.Post attribute), [13](#)

url (basc_py4chan.Thread attribute), [11](#)