BASC-py4chan Documentation

Release 0.4.2

Antonizoon (Lawrence Wu)

Jan 22, 2018

Contents

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!

CHAPTER 1

General Documentation

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, 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/thumbnail/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. `<br>` 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!

CHAPTER 2

API Documentation

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

name
string – Name of the board, such as “tg” or “etc”.

title
string – Board title, such as “Animu and Mango”.

is_worksafe
bool – Whether this board is worksafe.

page_count
int – How many pages this board has.

threads_per_page
int – How many threads there are on each page.

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
bool – Whether the thread has been closed.

sticky
bool – Whether this thread is a 'sticky'.

archived
bool – Whether the thread has been archived.

bumplimit
bool – Whether the thread has hit the bump limit.

imagelimit
bool – Whether the thread has hit the image limit.

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

topic
basc_py4chan.Post – Topic post of the thread, the OP.

posts
list of *basc_py4chan.Post* – List of all posts in the thread, including the OP.

all_posts
list of *basc_py4chan.Post* – List of all posts in the thread, including the OP and any omitted posts.

url
string – URL of the thread, not including semantic slug.

semantic_url
string – URL of the thread, with the semantic slug.

semantic_slug
string – The 'pretty URL slug' assigned to this thread by 4chan.

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
int – ID of this post. Eg: 123123123, 456456456.

poster_id
int – Poster ID.

name
string – Poster's name.

email
string – Poster's email.

tripcode
string – Poster's tripcode.

subject
string – Subject of this post.

comment
string – This comment, with the <wbr> tag removed.

html_comment
string – Original, direct HTML of this comment.

text_comment
string – Plaintext version of this comment.

is_op
bool – Whether this is the OP (first post of the thread).

spoiler
bool – Whether the attached file is spoiled.

timestamp
int – Unix timestamp for this post.

datetime
datetime.datetime – Datetime time of this post.

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

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

url
string – URL of this post.

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 *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
string – MD5 hash of the file attached to this post.

file_md5_hex
string – Hex-encoded MD5 hash of the file attached to this post.

filename
string – Name of the file attached to this post.

filename_original
string – Original name of the file attached to this post.

file_url
string – URL of the file attached to this post.

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

file_size
int – Size of the file attached to this post.

file_width
int – Width of the file attached to this post.

file_height
int – Height of the file attached to this post.

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

thumbnail_width
int – Width of the thumbnail attached to this post.

thumbnail_height
int – Height of the thumbnail attached to this post.

thumbnail_fname
string – Filename of the thumbnail attached to this post.

thumbnail_url
string – URL of the thumbnail attached to this post.

File objects are not instantiated directly, but through a *basc_py4chan.File* object with an attribute like *basc_py4chan.Post.first_file*.

Python Module Index

b

Index

T

U