sponsorblock.py

Release 0.2.1

Wasi Master

Jun 16, 2023

CONTENTS:

CHAPTER

ONE

API REFERENCE

If the API reference does not have the documentation for any method. See the source code, since all methods do have docstrings that you can use.

1.1 Submodules

1.1.1 sponsorblock.client module

The client module provides a Client for the SponsorBlock API.

Module for the client class.

class sponsorblock.client.**Client**(**args*, ***kwargs*)

Bases: object

A client for making requests to the sponsorblock server.

add_skip_segments(*video_id: str*, ***, *segment:* Segment | *None* = *None*, *segments: List[*Segment*]* | *None* = *None*, *service: str* = *'YouTube'*) → None

Add a skip segment to the server.

Parameters

- **video_id** (*str*) – The id of the video to add the skip segment to, can be a video url too.
- **segment** (Segment) – The skip segment to add
- **segments** (*List[*Segment*]*) – The list of skip segments to add
- **service** (*str*) – The service to use, default is 'YouTube'. See https://wiki.sponsor.ajay.app/w/Types#service.

Raises

- **ValueError** – No segments were specified
- **ValueError** – The video id is invalid
- ***InvalidJSONException*** – The server returned invalid JSON
- ***BadRequest*** – The server returned a 400 error, most likely because your inputs are wrong/impossible
- ***Forbidden*** – The server returned a 403 error, most likely because you are not allowed to add skip segments
- ***RateLimitException*** – The server returned a 429 error, most likely because you are making too many requests

- ***DuplicateException*** – The server returned a 409 error, most likely because the skip segment already exists
- ***ServerException*** – The server returned a 500 error, most likely because the server is down
- ***UnexpectedException*** – The server returned a response code that is not handled

Examples

```
>>> """Adding new segments to a YouTube video (Note: this is a example and don't␣
↪try this yourself)"""
>>> import sponsorblock as sb
>>> client = sb.Client()
>>> client.add_skip_segments(
        "https://www.youtube.com/watch?v=kJQP7kiw5Fk",
        segment=sb.Segment(category="music_offtopic", start=0, end=21.808434,␣
↪action_type="skip")
    )
```

get_saved_days_formatted()

get_saved_time_for_user()

get_segment_info()

get_skip_segments()

get_skip_segments_with_hash()

get_top_users()

get_total_stats()

get_user_info()

get_user_name()

get_views_for_user()

post_viewed_video_sponsor_time(*uuid:* Segment | *str*)

Notifies the server that a segment has been skipped.

Parameters

uuid (*Union[Segment, str]*) – The uuid of the segment that was skipped

Raises

- ***BadRequest*** – The server returned a 400 error, most likely because your inputs are wrong/impossible
- ***ServerException*** – The server returned a 500 error, most likely because the server is down
- ***UnexpectedException*** – The server returned a response that was not 200, 400, or 500

Examples

```
>>> import sponsorblock as sb
>>> client = sb.Client()
>>> segments = client.get_skip_segments("https://www.youtube.com/watch?
↪v=kJQP7kiw5Fk")
>>> client.post_viewed_video_sponsor_time(segments[1])
```

search_for_user()

set_user_name(*user_name: str*)

Sets the user name for the current user.

Parameters

user_name (*str*) – The user name to set for the current user.

Raises

- ***BadRequest*** – The server returned a 400 error, most likely because your inputs are wrong/impossible
- ***ServerException*** – The server returned a 500 error, most likely because the server is down
- ***UnexpectedException*** – The server returned a response that was not 200, 400, or 500

Examples

```
>>> import sponsorblock as sb
>>> client = sb.Client("your local user id")
>>> client.set_user_name("NoobMaster69")
```

vote_skip_segment(*uuid:* Segment | *str*, ***, *vote: str | int | bool | None = None*, *category: Literal['sponsor', 'selfpromo', 'interaction', 'intro', 'outro', 'preview', 'music_offtopic', 'poi_highlight', 'filler'] | None = None*) → None

Votes on a skip segment.

Parameters

- **uuid** (*Union[Segment, str]*) – segment or uuid of the segment being voted on
- **vote** (*Union[str, int, bool], optional*) – The vote to vote on the skip segment. Can be any of `yes`, `upvote`, `up`, `good`, `1`, `True` for upvoting, `no`, `downvote`, `down`, `bad`, `0`, `False` for downvoting, and `undo` for undoing a given vote
- **category** (*Category, optional*) – The category of the skip segment. This can be used as an alternative to the vote parameter, by default None

Raises

- `ValueError` – You passed both vote and category.
- ***BadRequest*** – The server returned a 400 error, most likely because your inputs are wrong/impossible
- ***Forbidden*** – The server returned a 403 error, most likely because you are not allowed to vote on the skip segment
- ***ServerException*** – The server returned a error, most likely because the server is down

- *UnexpectedException* – The server returned a response that was not 200, 400, 403, 429, 409, or 500

Examples

```
>>> import sponsorblock as sb
>>> client = sb.Client()
>>> segments = client.get_skip_segments("https://www.youtube.com/watch?
↪v=kJQP7kiw5Fk")
>>> client.vote_skip_segment(segments[0], 'yes')
```

sponsorblock.client.**raise_request_exception**(*response: Response*)

If the server returns a status code, which is not 200, it needs to raise an exception:

Every exception inherits from HTTPException

Parameters
response –

Returns

1.1.2 sponsorblock.models module

class sponsorblock.models.**SearchedUser**(*data*)

Bases: object

A user gotten by searching for his name

name

The name of the user

Type
str

id

The id of the user

Type
str

Warning: You should not make a instance of this yourself, this should only be created by the libray

class sponsorblock.models.**Segment**(*category: Literal['sponsor', 'selfpromo', 'interaction', 'intro', 'outro', 'preview', 'music_offtopic', 'poi_highlight', 'filler']*, *start: float | timedelta*, *end: float | timedelta*, *uuid: str | None = None*, *duration: timedelta | None = None*, *action_type: str | None = None*, ***, *data: dict | None = None*)

Bases: object

A skip segment

category

The category of the segment

Type
Category

`start`

The start time of the segment

Type
float

`end`

The end time of the segment

Type
float

`uuid`

The uuid of the segment, by default None

Type
Optional[str], optional

`duration`

The duration of the segment, by default None

Type
Optional[timedelta], optional

`action_type`

The action_type of the segment, by default None

Type
Optional[str], optional

`data`

The raw data that was used to create the segment, can be None if the segment was created manually

Type
Optional[dict], optional

Note: While creating your own instance you should only pass the category, start and end time. The other attributes won't do anything, those are only useful for segments gotten from the API.

`classmethod from_dict`(*data: dict*)

Generates a Segment object from a JSON dictionary.

Parameters
data (*dict*) – The dictionary containing the segment data.

Returns
The segment object gotten form the data.

Return type
Segment

Warning: This should not be used manually, this is for the library

class sponsorblock.models.**SegmentInfo**(*data*)

Bases: object

A class representing the segment info

video_id

The id of the video

Type
str

start_time

The start time of the segment

Type
float

end_time

The end time of the segment

Type
float

votes

The amount of votes the segment has

Type
int

locked

Whether the segment is locked

Type
int

uuid

The uuid of the segment

Type
str

user_id

The id of the user that created the segment

Type
str

time_submitted

The time the segment was submitted

Type
datetime.datetime

views

The amount of views the segment has

Type
int

`category`
: The category of the segment

Type
: str

`action_type`
: The action type of the segment

Type
: str

`service`
: The service of the segment

Type
: str

`video_duration`
: The duration of the video

Type
: float

`hashed_video_id`
: The hashed video id of the video

Type
: str

`user_agent`
: The user agent of the segment

Type
: str

> **Warning:** You should not make a instance of this yourself, this should only be created by the libray

`class sponsorblock.models.TopUser(user_name: str, view_count: int, total_submissions: int, minutes_saved: float)`

Bases: `object`

A top user.

`user_name`
: The name of the user

Type
: str

`view_count`
: The total number of views of the user's segments

Type
: int

`total_submissions`
: The total number of submissions the user has made

Type
int

minutes_saved

The amount of time the user has saved

Type
float

> **Warning:** You should not make a instance of this yourself, this should only be created by the libray

class sponsorblock.models.**TotalStats**(*data*)

Bases: object

The total stats

user_count

The amount of users, only available if count_contributing_users was true

Type
int

active_users

Sum of public install stats from Chrome webstore and Firefox addons store

Type
int

api_users

48-hour active API users

Type
int

view_count

The total number of views

Type
int

total_submissions

The total number of submissions

Type
int

minutes_saved

The total amount of time saved

> **Warning:** You should not make a instance of this yourself, this should only be created by the libray

class sponsorblock.models.**User**(*data: dict*)

Bases: object

A user

`user_id`
The id of the user

Type
str

`user_name`
The name of the user

Type
str

`minutes_saved`
The amount of minutes saved by the user

Type
int

`segment_count`
The amount of segments created by the user

Type
int

`ignored_segment_count`
The amount of segments that were ignored by the user

Type
int

`view_count`
The amount of views by the user

Type
int

`ignored_view_count`
The amount of views that were ignored by the user

Type
int

`warnings`
The amount of warnings the user has

Type
int

`reputation`
The amount of reputation the user has

Type
int

`vip`
Whether the user is a VIP

Type
bool

last_segment_id

The id of the last segment created by the user

Type

str

Warning: You should not make a instance of this yourself, this should only be created by the libray

1.1.3 sponsorblock.errors module

exception sponsorblock.errors.**BadRequest**(*message*, *response: Response*)

Bases: *HTTPException*

Raised when the status code is 400.

exception sponsorblock.errors.**DuplicateException**(*message*, *response: Response*)

Bases: *HTTPException*

Raised when the status code is 409.

exception sponsorblock.errors.**Forbidden**(*message*, *response: Response*)

Bases: *HTTPException*

Raised when the status code is 403.

exception sponsorblock.errors.**HTTPException**(*message*, *response: Response*)

Bases: Exception

Raised when the server returns an error code

exception sponsorblock.errors.**InvalidJSONException**(*message*, *response*)

Bases: Exception

Raised when the JSON gotten from the server is invalid

exception sponsorblock.errors.**NotFoundException**(*message*, *response: Response*)

Bases: *HTTPException*

Raised when the status code is 404.

exception sponsorblock.errors.**RateLimitException**(*message*, *response: Response*)

Bases: *HTTPException*

Raised when the status code is 429.

exception sponsorblock.errors.**ServerException**(*message*, *response: Response*)

Bases: *HTTPException*

Raised if the status code is bigger than 500.

exception sponsorblock.errors.**UnexpectedException**(*message*, *response: Response*)

Bases: *HTTPException*

Raised when an unknown error has occurred.

1.1.4 sponsorblock.utils module

class sponsorblock.utils.**Singleton**

Bases: type

class sponsorblock.utils.**SortType**(*value*)

Bases: Enum

0 for by minutes saved, 1 for by view count, 2 for by total submissions

See also:

sponsorblock.client.Client.get_top_users
Should be used with the SortType

MINUTES_SAVED = 0

TOTAL_SUBMISSIONS = 2

VIEW_COUNT = 1

sponsorblock.utils.**cache**(***kwargs*)

Custom cache implementation taken from https://stackoverflow.com/a/67555155/13123877.

Parameters

- **ttl** (*timedelta, optional*) – The time to live for the cache. Defaults to max time supported by the platform.
- **max_entries** (*int*) – The maximum number of entries to store in the cache..

sponsorblock.utils.**set_env_var**(*key: str*, *value: str*)

Sets an environment variable to the given value.

Parameters

- **key** (*str*) – The key of the environment variable to set.
- **value** (*str*) – The value of the environment variable to set.

Warning: Most of the time this shouldn't be used manually, but it's here in case you need to set the SPONSORBLOCK_USER_ID environment variable.

sponsorblock.py is a easy to use, fast, and powerful wrapper around the sponsorblock api

Example of getting the segments of a video using sponsorblock.py

```
>>> import sponsorblock as sb
>>> client = sb.Client()
>>> segments = client.get_skip_segments("https://www.youtube.com/watch?v=kJQP7kiw5Fk")
>>> segments
[
   Segment(category=music_offtopic, start=0, end=21.808434,␣
↪uuid=728cbf1743f4b5230ee4a9c7b254e316aa90720ec35297b17aaf6d23907c1a83,␣
↪duration=0:00:21.808434, action_type=skip),
   Segment(category=music_offtopic, start=249.6543, end=281.521,␣
↪uuid=ae38abe70c63b093eaeb1c2c437aa3275856646c326ee23b5ff70dcb4190c92f,␣
```

(continues on next page)

(continued from previous page)

```
↪duration=0:00:31.866700, action_type=skip),
   Segment(category=outro, start=274.674, end=281.521,␣
↪uuid=cd335e7f406df63e460b4b02db71cc57344529d381bb9fc482960f338ff4ae37,␣
↪duration=0:00:06.847000, action_type=skip)
]
>>> segments[0].category
'music_offtopic'
>>> segments[1].start, segments[1].end
(249.6543, 281.521)
>>> segments[2].duration.seconds
6
```

For more information, see *API Reference*

There is also a cli that you can use to get segments from the command line (beta). To use that, run:

```
sponsorblock video_id
```

and pass your desired video_id.

CHAPTER

TWO

INDICES AND TABLES

- genindex
- modindex
- *API Reference*
- search

PYTHON MODULE INDEX

S

INDEX