hubspot3 Documentation

Release 3.2.53

Jacobi Petrucciani

Jun 15, 2023

Contents

A python wrapper around HubSpot's APIs, for *python 3.7+*

hubspot3's source code hosted on GitHub.

New to hubspot3? These may help:

CHAPTER 1

Quickstart

This document presents a brief, high-level overview of hubspot3's primary features.

hubspot3 is a python wrapper around HubSpot's APIs, for python 3.

Note: Be aware that this uses the HubSpot API directly, so you are subject to all of the guidelines that HubSpot has in place.

At the time of writing, HubSpot has the following limits in place for API requests:

- 10 requests per second
- 40,000 requests per day. This daily limit resets at midnight based on the time zone setting of the HubSpot account

1.1 Installation

```
# install hubspot3
pip install hubspot3
```

1.2 Basic Usage

```
from hubspot3 import Hubspot3

API_KEY = "your-api-key"

client = Hubspot3(api_key=API_KEY)

# all of the clients are accessible as attributes of the main Hubspot3 Client
```

(continues on next page)

(continued from previous page)

```
contact = client.contacts.get_contact_by_email('testingapis@hubspot.com')
contact_id = contact['vid']

all_companies = client.companies.get_all()

# new usage limit functionality - keep track of your API calls
client.usage_limits
# <Hubspot3UsageLimits: 28937/1000000 (0.028937%) [reset in 22157s, cached for 299s]>

client.usage_limits.calls_remaining
# 971063
```

1.3 Individual Clients

```
from hubspot3.companies import CompaniesClient

API_KEY = "your-api-key"

client = CompaniesClient(api_key=API_KEY)

for company in client.get_all():
    print(company)
```

1.4 Passing Params

```
import json
from hubspot3.deals import DealsClient

deal_id = "12345"
API_KEY = "your_api_key"

deals_client = DealsClient(api_key=API_KEY)

params = {
    "includePropertyVersions": "true"
}  # Note values are camelCase as they appear in the Hubspot Documentation!

deal_data = deals_client.get(deal_id, params=params)
print(json.dumps(deal_data))
```

CHAPTER 2

Extending

Some of the APIs are not yet complete!

If you'd like to use an API that isn't yet in this repo, you can extend the BaseClient class! Also, feel free to make a PR if you think it would help others!

2.1 Extending from the BaseClient

The following code block shows an example (provided by guysoft)

```python
import json
from hubspot3.base import BaseClient


PIPELINES_API_VERSION = "1"


class PipelineClient(BaseClient):
    """
    Lets you extend to non-existing clients, this example extends pipelines
    """

    def __init__(self, *args, **kwargs):
        super(PipelineClient, self).__init__(*args, **kwargs)

    def get_pipelines(self, **options):
        params = {**options}  # unpack options as params for the api call

        return self._call("pipelines", method="GET", params=params)

    def _get_path(self, subpath):
        return f"deals/v{self.options.get('version') or PIPELINES_API_VERSION}/
↪{subpath}"
```

(continues on next page)

(continued from previous page)

```python


if __name__ == "__main__":
    API_KEY = "your_api_key"
    a = PipelineClient(api_key=API_KEY)
    print(json.dumps(a.get_pipelines()))
```

2.2 Working with Pagination

Pagination can be tricky with how the hubspot API is set up.

Below is an example of how to deal with pagination from the DealsClient included in this library, specifically to get x number of recently created deals.

```python
def get_recently_created(
    self, limit=100, offset=0, since=None, include_versions=False, **options
):
    """
    get recently created deals
    up to the last 30 days or the 10k most recently created records

    since: must be a UNIX formatted timestamp in milliseconds
    """
    finished = False
    output = []
    query_limit = 100  # max according to the docs

    while not finished:
        params = {
            "count": query_limit,
            "offset": offset,
            "includePropertyVersions": include_versions,
        }
        if since:
            params["since"] = since
        batch = self._call(
            "deal/recent/created",
            method="GET",
            params=params,
            doseq=True,
            **options
        )
        output.extend(
            [
                prettify(deal, id_key="dealId")
                for deal in batch["results"]
                if not deal["isDeleted"]
            ]
        )
        finished = not batch["hasMore"] or len(output) >= limit
        offset = batch["offset"]

    return output[:limit]
```

CHAPTER 3

Command-line interface

The hubspot3 client comes with an optional command-line interface that makes it easy to use the client's features without necessarily importing it within a Python project.

3.1 Installation

To use the command-line interface you will have to install an additional Python requirement:

```
pip install hubspot3[cli]
```

3.2 Usage

After the installation you can use the client with the `hubspot3` command, e.g. to display the help:

```
hubspot3 --help
```

This will display all arguments for the general usage of the command-line client, like the API key (or a config file, see below). Please take a look into the constructor of the `hubspot3.Hubspot3` class for the list of provided arguments.

By just calling `hubspot3` you will get a list of all featured APIs that are available, e.g.:

```
huspot3

Usage:       hubspot3 -
             hubspot3 - blog
             hubspot3 - blog-comments
             hubspot3 - broadcast
[...]
```

By attaching the API name you can access the specific API endpoints:

```
hubspot3 deals

Usage:       hubspot3 deals
             hubspot3 deals associate
             hubspot3 deals create
             hubspot3 deals get
[...]
```

Each API method has it's own documentation and the help can be displayed like this:

```
hubspot3 deals get -- --help

[...]
Docstring:   Supported ARGS/KWARGS are:
DEAL_ID [--OPTIONS ...]
--deal-id DEAL_ID [--OPTIONS ...]
[...]
```

This will display the methods parameter list and how to pass them to the specific API endpoint, in this case you can get the deal information for a specific deal with:

```
hubspot3 --api-key xxxxxxxxx-xxxx-xxx-xxxx-xxxxx deals get --deal-id 100
```

Note: You will notice the leading `--` in front of the `--help` parameter. Hubspot3 uses the `python-fire` library to generate a dynamic command-line interface for all API clients. If you don't add the `--` the help command will be passed as an argument API method instead of invoking Fire's help output, so we have to set it. Generally speaking, the arguments for the API method call need to be separated from general CLI arguments using `--`.

3.2.1 Configuration file

Instead of providing the API key (`--api-key`) or other settings (like `--client-id` or `--timeout`) as parameters you can also create a local JSON file, that contains all the settings you want to pass to the client:

```
{
    "api_key": "xxxxxxxxx-xxxx-xxx-xxxx-xxxxx",
    "timeout": 60
}
```

Simply call the hubspot client with the `--config` parameter:

```
hubspot3 --config config.json
```

3.2.2 Using `stdin` for parameters

Some of the data you want to pass to the hubspot3 may be sensitive or just too much to pass it as a regular parameter. Therefore you can simply pass data from `stdin` to the client so that the data can be streamed and won't occur in your shell history. To do so just use the token `__stdin__` for one of your parameters:

```
hubspot3 --config config.json \
   contacts update --contact_id 451 \
   --data "__stdin__" < contact_data.json
```

In this case `contact_data.json` is a JSON file that contains the contacts data to update:

```
{
    "properties": [
        {
            "property": "firstname",
            "value": "Adrian"
        },
        {
            "property": "lastname",
            "value": "Mott"
        }
    ]
}
```

3.3 Extending the APIs

There is one specialty in the way python-fire discovers the API clients: it will parse all classes that are derived from `BaseClient` and are provided as a property within the `hubspot3.Hubspot3` class. Within these API clients python-fire will look for public methods and provide them as a command-line operation.

If you want to hide python-fire certain properties from the `hubspot3.Hubspot3` class (e.g. because it will instantly make a call to Hubspot or the property doesn't reflect an API endpoint) you can hide that property by extending the `Hubspot3CLIWrapper.IGNORED_PROPERTIES` tuple within `hubspot3/__main__.py`:

```
class Hubspot3CLIWrapper(object):

    IGNORED_PROPERTIES = ('me', 'usage_limits', 'my_method_to_hide')
```

In a similar fashion, public methods of the individual API clients can be hidden by extending the dictionary `ClientCLIWrapper.IGNORED_METHODS` within the same module. It uses the client classes as keys and iterables containing method names to hide as values:

```
class ClientCLIWrapper(object):

    IGNORED_METHODS = {
        LeadsClient: ('camelcase_search_options',),
        MyClient: ('my_method',),
    }
```

CHAPTER 4

Note

If you find any bugs, odd behavior, or have an idea for a new feature please don’t hesitate to open an issue!

CHAPTER 5

Indices and tables

- genindex
- modindex
- search