Mopinion Client

Mopinion

Jan 22, 2021

CONTENTS:

mopinion is a python client that provides functionality for authentication, authorization and requesting resources. It comes with an easy, beautiful and elegant way of interacting with our Mopinion Data API.

CHAPTER

ONE

ABOUT

This package was developed by Mopinion to facilitate end-users interacting with the API.

CHAPTER

TWO

CONTENTS

2.1 Installation

2.1.1 Requirements

- requests

2.1.2 Windows (pip)

The following recipe is still a work in progress:

1. Install Python 3.* (stable)
2. Start the command prompt
3. Install mopinion:

```
pip install mopinion
```

Note: You might need to setup your C++ compiler according to this.

2.1.3 Advanced: local setup with system Python (Ubuntu)

These instructions make use of the system-wide Python 3 interpreter:

```
$ sudo apt install python3-pip
```

Install mopinion:

```
$ pip install --user mopinion
```

2.1.4 Advanced: local setup for development (Ubuntu)

These instructions assume that `git`, `python3`, `pip`, and `virtualenv` are installed on your host machine.

Clone the mopinion-python-api repository:

```
$ git clone https://github.com/mopinion/mopinion-python-api
```

Create and activate a virtualenv:

```
$ cd mopinion-python-api
$ virtualenv --python=python3 .venv
$ source .venv/bin/activate
```

Run the tests:

```
(.venv) $ pytest
```

2.2 Quickstart

This is a quick introduction, for a complete guide please go to *Mopinion Client* or *Requesting Resources*.

2.2.1 Instantiating the MopinionCLient

Credentials can be created via the Mopinion Suite at Integrations » Feedback API in classic interface or in the Raspberry interface, provided your package includes API access.

You can also take a look at this link with the steps to get `private_key` and `public_key`

```
>>> from mopinion_api import APIClient
>>> client = APIClient(public_key=PUBLIC_KEY, private_key=PRIVATE_KEY)
```

2.2.2 Checking for availability

```
>>> assert client.is_available()
```

2.2.3 Making a request

Request your account.

```
>>> response = client.resource("account")
>>> assert response.json()["_meta"]["code"] == 200
```

Or request deployments.

```
>>> response = client.resource("deployments")
>>> assert response.json()["_meta"]["code"] == 200
```

If you need further examples about requesting resources please go to *Mopinion Client* or *Requesting Resources*.

2.3 Mopinion Client

The intention of developing a MopinionClient is to make easy, beautiful and elegant when interacting with our API.

Credentials can be created via the Mopinion Suite at Integrations » Feedback API in classic interface or in the Raspberry interface, provided your package includes API access.

You can also take a look at this link with the steps to get `private_key` and `public_key`

2.3.1 MopinionClient Specifications

class mopinion_client.**MopinionClient**(*public_key: str*, *private_key: str*, *max_retries: int = 3*)

Client to interact with Mopinion API.

Provides functionality for authentication, authorization and requesting resources. Steps during instantiation:

1. Credential validations.
2. Instantiation of a session object from `requests.Session` that will be used in each request.
3. Retrieval of `signature_token` from the API for an specific `private_key` and `public_key`.

When instantiating, a signature token is retrieved from the API and stored in the `signature_token` attribute using your `private_key` and `public_key`. The `signature_token` will be used in each request.

In each request, a HMAC signature will be created using SHA256-hashing, and encrypted with your `signature_token`. This HMAC signature is encode together with the `public_key`. After this encryption, the xtoken is set into the headers under the `X-Auth-Token` key.

Parameters

- **public_key** (*str*) –
- **private_key** (*str*) –
- **max_retries** (*int*) – Defaults to 3.

is_available(*verbose: bool = False*) → Union[dict, bool]

Test the API's availability.

It return a boolean `True`/`False` in case the API is available or not. In case we need extra information about the state of the API, we can provide a flag `verbose=True`.

Examples

```
>>> from mopinion_client import MopinionClient
>>> client = MopinionClient(public_key=PUBLICKEY, private_key=PRIVATEKEY)
>>> assert client.is_available()
>>> r = client.is_available(verbose=True)
>>> assert r["code"] == 200 and r["response"] == "pong" and r["version"] ==
↪"2.0.0"
```

request(*endpoint: str*, *method: str = 'GET'*, *version: str = '2.0.0'*, *verbosity: str = 'normal'*, *content_negotiation: str = 'application/json'*, *body: Optional[dict] = None*, *query_params: Optional[dict] = None*) → requests.models.Response

Generic method to send requests to our API.

Wrapper on top of `requests.Session.request` method adding token encryption on headers. Everytime we call *request* five steps are applied:

1. Validation of arguments.
2. Token creation - token depends on *endpoint* argument and *signature_token*.
3. Preparation of parameter dictionary. Add token to headers.
4. Make request.
5. Return response.

Parameters

- **endpoint** (*str*) – API endpoint.
- **method** (*str*) – HTTP Method.
- **version** (*str*) – API Version.
- **verbosity** (*str*) – *normal*, *quiet* or *full*.
- **content_negotiation** (*str*) – *application/json* or *application/x-yaml*.
- **body** (*dict*) – Optional.
- **query_params** (*dict*) – Optional.

Returns response (requests.models.Response).

Examples

```
>>> from mopinion_client import MopinionClient
>>> client = MopinionClient(public_key=PUBLICKEY, private_key=PRIVATEKEY)
>>> response = client.request("/account")
>>> assert response.json()["_meta"]["code"] == 200
>>> response = client.request(endpoint="/deployments")
>>> assert response.json()["_meta"]["code"] == 200
>>> body = {"key": "key", "name": "My Test Deployment"},
>>> response = client.request(endpoint="/deployments", method="POST",␣
↪body=body)
>>> assert response.json()["_meta"]["code"] == 201
>>> endpoint = "/deployments/abt34")
>>> response = client.request(endpoint, method="DELETE")
>>> assert response.json()["_meta"]["code"] == 200
```

resource(*resource_name: str*, *resource_id: Optional[Union[str, int]] = None*, *sub_resource_name: Optional[str] = None*, *sub_resource_id: Optional[Union[str, int]] = None*, *method: str = 'GET'*, *version: str = '2.0.0'*, *verbosity: str = 'normal'*, *content_negotiation: str = 'application/json'*, *query_params: Optional[dict] = None*, *body: Optional[dict] = None*, *iterator: bool = False*) → Union[requests.models.Response, collections.abc.Iterator]

Method to send requests to our API.

Abstraction of `mopinion_api.MopinionClient.request`. Interacts with the API in term of resources and subresources, and also, enables iterator protocol when requesting large resources.

Parameters

- **resource_name** (*str*) –
- **resource_id** (*str/int*) –
- **sub_resource_name** (*str*) –
- **sub_resource_id** (*str*) –

- **method** (*str*) – HTTP Method.
- **version** (*str*) – API Version.
- **verbosity** (*str*) – *normal*, *quiet* or *full*.
- **content_negotiation** (*str*) – *application/json* or *application/x-yaml*.
- **body** (*dict*) – Optional.
- **query_params** (*dict*) – Optional.
- **iterator** (*bool*) – If sets to *True* an iterator will be returned.

Returns response (requests.models.Response) or iterator (collections.abc.Iterator)

The endpoint is built from `mopinion_api.dataclasses.ResourceUri` and the parameters are:

- resource_name (str) Required
- resource_id (int/str) Optional
- subresource_name (str) Optional
- subresource_id (str) Optional

Resources and sub-resources options:

- The `resource_name` options are: "account", "deployments", "datasets", "reports".
- The `subresource_name` options are: "fields", "feedback".

You can also use the constants defined in the `mopinion_api.MopinionClient` class.

- The `resource_name` options are: `RESOURCE_ACCOUNT`, `RESOURCE_DEPLOYMENTS`, `RESOURCE_DATASETS`, `RESOURCE_REPORTS`.
- The `subresource_name` options are: `SUBRESOURCE_FIELDS`, `SUBRESOURCE_FEEDBACK`.

Examples

```
>>> from mopinion_client import MopinionClient
>>> client = MopinionClient(public_key=PUBLICKEY, private_key=PRIVATEKEY)
>>> response = client.resource("account")
>>> assert response.json()["_meta"]["code"] == 200
>>> response = client.resource(resource_name=client.RESOURCE_ACCOUNT)  # same␣
↪as above
>>> assert response.json()["_meta"]["code"] == 200
```

When working with the API there is a limit of elements retrieved. The `limit` parameters defaults to *10*. You can increase the limit, or you can request resources using the flag `generator=True`. This returns a Generator which traverses these pages for you and yields each result in the current page before retrieving the next page.

Examples

```
>>> from mopinion_client import MopinionClient
>>> client = MopinionClient(public_key=PUBLICKEY, private_key=PRIVATEKEY)
>>> iterator = client.resource("account", iterator=True)
>>> response = next(iterator)
>>> assert response.json()["_meta"]["code"] == 200
```

Below some more examples.

Examples

```
>>> from mopinion_client import MopinionClient
>>> client = MopinionClient(public_key=PUBLICKEY, private_key=PRIVATEKEY)
>>> response = client.resource("account")
>>> assert response.json()["_meta"]["code"] == 200
>>> response = client.resource("deployments")
>>> assert response.json()["_meta"]["code"] == 200
>>> body={"key": "mydeploymentkey3", "name": "My Test Deployment"},
>>> response = client.resource("deployments", method="POST", body=body)
>>> assert response.json()["_meta"]["code"] == 201
>>> response = client.resource("deployments", resource_id="abt34", method=
↪"DELETE")
>>> assert response.json()["_meta"]["code"] == 200
```

2.4 Requesting Resources

The next examples follow the order from the API documentation.

Credentials can be created via the Mopinion Suite at Integrations » Feedback API in classic interface or in the Raspberry interface, provided your package includes API access.

You can also take a look at this link with the steps to get `private_key` and `public_key`

2.4.1 General

After instalation, open a python terminal and set the `public_key`, and `private_key`.

```
>>> import os
>>> from mopinion_client import MopinionClient
>>> PUBLIC_KEY = os.environ.get("PUBLIC_KEY")
>>> PRIVATE_KEY = os.environ.get("PRIVATE_KEY")
>>> SIGNATURE_TOKEN = os.environ.get("SIGNATURE_TOKEN")

A token signature is retrieved from the API and set to
>>> from mopinion_client import MopinionClient
>>> PUBLIC_KEY = os.environ.get("PUBLIC_KEY")
>>> PRIVATE_KEY = os.environ.get("PRIVATE_KEY")
>>> SIGNATURE_TOKEN = os.environ.get("SIGNATURE_TOKEN")
```

A token signature is retrieved from the API and set to `signature_token`.

```
>>> client = MopinionClient(public_key=PUBLIC_KEY, private_key=PRIVATE_KEY)
>>> assert SIGNATURE_TOKEN == client.signature_token  # client requests the signature␣
↪token
```

To see the availability of the API you can call `is_available()`.

```
>>> assert client.is_available()
>>> r = client.is_available(verbose=True)
>>> assert r["code"] == 200 and r["response"] == "pong" and r["version"] == "2.0.0"
```

2.4.2 Examples with `mopinion_client.MopinionClient.resource`

This set of examples use the method `resource` from the `MopinionClient`.

Resource Account

Get your account.

```
>>> response = client.resource(resource_name=client.RESOURCE_ACCOUNT)
>>> assert response.json()["_meta"]["code"] == 200
>>> print(response.json())
{'name': 'Mopinion', 'package': 'Growth', 'enddate': '2021-02-13 00:00:00', 'number_
↪users': 10, ...
```

Get your account in yaml format.

```
>>> import yaml
>>> response = client.resource("account", content_negotiation=client.CONTENT_YAML)
>>> r = yaml.safe_load(response.text)
>>> assert r["_meta"]["code"] == 200
```

When requesting with `verbosity='quiet'` no `_meta` info is returned.

```
>>> response = client.resource("account", verbosity=client.VERBOSITY_QUIET)
>>> assert "_meta" not in response.json()
```

Resource Deployments

Getting deployments.

```
>>> response = client.resource(resource_name=client.RESOURCE_DEPLOYMENTS)
>>> assert response.json()["_meta"]["code"] == 200
>>> response.json()
{'0': {'key': 'defusvnns6mkl2vd3wc0wgcjh159uh3j', 'name': 'Web Feedback Deployment'},
↪'_meta':...
```

Add a new deployment to your account.

```
>>> body = {"key": "key", "name": "My Test Deployment"}
>>> response = client.resource("deployments", method="POST", body=body)
>>> assert response.json()["_meta"]["code"] == 201
>>> response.json()
{'key': 'key', 'name': 'My Test Deployment', '_meta': {'co...
```

Deleting a deployment.

```
>>> response = client.resource(client.RESOURCE_DEPLOYMENTS, "abt34", method="DELETE")
>>> assert response.json()["_meta"]["code"] == 200
>>> response.json()
{'executed': True, 'resources_affected': {'deployments': ['mydeploymentk...
```

Resource Datasets

Getting a dataset.

```
>>> response = client.resource(resource_name=client.RESOURCE_DATASETS, resource_
↪id=1234)
>>> assert response.json()["_meta"]["code"] == 200
```

Updating a dataset.

```
>>> body = {"name": "My updated name", "description": "My updated description"}
>>> response = client.resource("datasets", resource_id=1234, method="PUT", body=body)
>>> assert response.json()["_meta"]["code"] == 200
```

Deleting a dataset.

```
>>> response = client.resource("datasets", resource_id=1234, method="DELETE")
>>> assert response.json()["_meta"]["code"] == 200
```

Add a new dataset to a report.

```
>>> body = {"name": "Web care performance", "report_id": "854", "description":
↪"Historic data import"}
>>> response = client.resource("datasets", method="POST", body=body)
>>> assert response.json()["_meta"]["code"] == 201
```

Get fields for a dataset.

```
>>> response = client.resource("datasets", 1234, "fields")
>>> assert response.json()["_meta"]["code"] == 200
```

Resource Fields

Get fields for a dataset.

```
>>> response = client.resource("datasets", 1234, "fields")
>>> assert response.json()["_meta"]["code"] == 200
```

Get fields for a report.

```
>>> response = client.resource("reports", 1234, "fields")
>>> assert response.json()["_meta"]["code"] == 200
```

Resource Feedback

Get feedback from a dataset.

```
>>> response = client.resource("datasets", 1234, "feedback", "abt34")
>>> assert response.json()["_meta"]["code"] == 200
```

Get feedback for a report.

```
>>> response = client.resource("reports", 1234, "feedback", "abt34")
>>> assert response.json()["_meta"]["code"] == 200
```

Resource Reports

Get some basic info on a report.

```
>>> response = client.resource("reports", 1234)
>>> assert response.json()["_meta"]["code"] == 200
```

Update an existing report.

```
>>> body = {"name": "Customer Support", "description": "Support related", "language":
↪"en_US"}
>>> response = client.resource("reports", resource_id=1234, method="PUT", body=body)
>>> assert response.json()["_meta"]["code"] == 200
```

And deleting a dataset.

```
>>> response = client.resource("reports", resource_id=1234, method="DELETE")
>>> assert response.json()["_meta"]["code"] == 200
```

Add a new report to the account.

```
>>> body = {"name": "Customer Support", "description": "Support related", "language":
↪"en_US"}
>>> response = client.resource("reports", method="POST", body=body)
>>> assert response.json()["_meta"]["code"] == 201
```

2.4.3 Examples with `mopinion_client.MopinionClient.request`

This set of examples use the method `request` from the `MopinionClient`.

Resource Account

Get your account.

```
>>> response = client.request("/account")
>>> assert response.json()["_meta"]["code"] == 200
>>> print(response.json())
{'name': 'Mopinion', 'package': 'Growth', 'enddate': '2021-02-13 00:00:00', 'number_
↪users': 10, ...
```

Get your account in yaml format.

```
>>> import yaml
>>> response = client.request("/account", content_negotiation=client.CONTENT_YAML)
>>> r = yaml.safe_load(response.text)
>>> assert r["_meta"]["code"] == 200
```

When requesting with `verbosity='quiet'` no `_meta` info is returned.

```
>>> response = client.request("/account", verbosity=client.VERBOSITY_QUIET)
>>> assert "_meta" not in response.json()
```

Resource Deployments

Getting deployments.

```
>>> response = client.request("/deployments")
>>> assert response.json()["_meta"]["code"] == 200
>>> response.json()
```

Add a new deployment to your account.

```
>>> body = {"key": "key", "name": "My Test Deployment"}
>>> response = client.request("/deployments", method="POST", body=body)
>>> assert response.json()["_meta"]["code"] == 201
>>> response.json()
```

Deleting a deployment.

```
>>> response = client.request("/deployments/abt34", method="DELETE")
>>> assert response.json()["_meta"]["code"] == 200
>>> response.json()
```

Resource Datasets

Getting a dataset.

```
>>> response = client.request("/datasets/1234")
>>> assert response.json()["_meta"]["code"] == 200
```

Updating a dataset.

```
>>> body = {"name": "My updated name", "description": "My updated description"}
>>> response = client.request("/datasets/1234", method="PUT", body=body)
>>> assert response.json()["_meta"]["code"] == 200
```

Deleting a dataset.

```
>>> response = client.request("/datasets/1234", method="DELETE")
>>> assert response.json()["_meta"]["code"] == 200
```

Add a new dataset to a report.

```
>>> body = {"name": "Web care performance", "report_id": "854", "description":
↪"Historic data import"}
>>> response = client.request("/datasets", method="POST", body=body)
>>> assert response.json()["_meta"]["code"] == 201
```

Get fields for a dataset.

```
>>> response = client.request("/datasets/1234/fields")
>>> assert response.json()["_meta"]["code"] == 200
```

Resource Fields

Get fields for a dataset.

```
>>> response = client.request("/datasets/1234/fields")
>>> assert response.json()["_meta"]["code"] == 200
```

Get fields for a report.

```
>>> response = client.request("/reports/1234/fields")
>>> assert response.json()["_meta"]["code"] == 200
```

Resource Feedback

Get feedback from a dataset.

```
>>> response = client.request("datasets/1234/feedback/abt34")
>>> assert response.json()["_meta"]["code"] == 200
```

Get feedback for a report.

```
>>> response = client.request("reports/1234/feedback/abt34")
>>> assert response.json()["_meta"]["code"] == 200
```

Resource Reports

Get some basic info on a report.

```
>>> response = client.request("/reports/1234")
>>> assert response.json()["_meta"]["code"] == 200
```

Update an existing report.

```
>>> body = {"name": "Customer Support", "description": "Support related", "language":
↪"en_US"}
>>> response = client.request("/reports/1234", method="PUT", body=body)
>>> assert response.json()["_meta"]["code"] == 200
```

And deleting a dataset.

```
>>> response = client.resource("reports/1234", method="DELETE")
>>> assert response.json()["_meta"]["code"] == 200
```

Add a new report to the account.

```
>>> body = {"name": "Customer Support", "description": "Support related", "language":
↪"en_US"}
>>> response = client.resource("/reports", method="POST", body=body)
>>> assert response.json()["_meta"]["code"] == 201
```

2.4.4 Examples with the iterator

When working with the API there is a limit of elements retrieved. The `limit` parameters defaults to *10*. You can increase the limit, or you can request resources using the flag `generator=True`. This returns a Generator which traverses these pages for you and yields each result in the current page before retrieving the next page.

```
>>> from mopinion_client import MopinionClient
    >>> client = MopinionClient(public_key=PUBLICKEY, private_key=PRIVATEKEY)
    >>> iterator = client.resource("deployments", iterator=True)
    >>> response = next(iterator)
    >>> assert response.json()["_meta"]["code"] == 200

Also, for example, requesting fields for a report.

    >>> iterator = client.resource("datasets", 1234, "fields", iterator=True)
    >>> response = next(iterator)
    >>> assert response.json()["_meta"]["code"] == 200
>>> client = MopinionClient(public_key=PUBLICKEY, private_key=PRIVATEKEY)
>>> iterator = client.resource("deployments", iterator=True)
>>> response = next(iterator)
>>> assert response.json()["_meta"]["code"] == 200
```

Also, for example, requesting fields for a report.

```
>>> iterator = client.resource("datasets", 1234, "fields", iterator=True)
>>> response = next(iterator)
>>> assert response.json()["_meta"]["code"] == 200
```

2.5 License

2.6 Any help

The Mopinion Python Client API is maintained by Mopinion Development Team. Everyone is encouraged to file bug reports, feature requests, and pull requests through GitHub. For more information please email our Support Team at support@mopinion.com.

CHAPTER

THREE

INDICES AND TABLES

- genindex
- modindex
- search

PYTHON MODULE INDEX

m

INDEX