Mopinion Client

Mopinion

Jul 12, 2023

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

CHAPTER

ONE

ABOUT MOPINION

Mopinion is a leading all-in-one user feedback platform that helps digital enterprises listen, understand, and act across all digital touchpoints (web, mobile, and email). Join some of the most forward-thinking digital teams from companies such as T-mobile, eBay, TSB Bank, Walmart, Hotels.com, Decathlon, Ahold, Mediacorp Ltd, and many more.

Please visit the website for more information about the product at mopinion.com

1.1 Installation

1.1.1 Requirements

- requests

1.1.2 Windows (pip)

1. Install Python>=3.6 (stable)
2. Start the command prompt
3. Install mopinion:

```
pip install mopinion
```

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

1.1.3 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
```

1.1.4 Advanced: local setup for development (Ubuntu)

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

Clone the mopinion-python-api repository:

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

Create and activate a virtualenv:

```
$ cd mopinion-python-api
$ pipenv install --python=python3
$ pipenv shell
```

Run the tests:

```
(mopinion-python-api) $ pytest
```

1.2 Quickstart

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

1.2.1 Instantiating the MopinionCLient

Credentials can be created via the Mopinion Suite at Integrations » Feedback API in the 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 a `private_key` and a `public_key`.

```
>>> from mopinion import MopinionClient
>>> client = MopinionClient(public_key=YOUR_PUBLIC_KEY, private_key=YOUR_PRIVATE_KEY)
```

1.2.2 Checking for availability

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

1.2.3 Making a request

Request your account.

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

Or request deployments.

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

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

1.3 Mopinion Client

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

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

Take a look at this link with the steps to get a `private_key` and a `public_key`.

1.3.1 MopinionClient Specifications

class mopinion.**MopinionClient**(*public_key: str*, *private_key: str*, *max_retries: int = 3*, *backoff_factor: int = 1*, *version: Optional[str] = None*, *verbosity: str = 'normal'*, *content_negotiation: str = 'application/json'*)

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

Parameters

- **public_key** (*str*) –
- **private_key** (*str*) –
- **verbosity** (*str*) – Defaults to normal.
- **version** (*str*) – If no version provided, default to latest.
- **content_negotiation** (*str*) – Defaults to application/json.
- **max_retries** (*int*) – Defaults to 3.
- **backoff_factor** (*int*) – Defaults to 1.

build_token(*endpoint: EndPoint*) → bytes

Get token

`get_account`(***kwargs*)

Get your account.

Parameters
kwargs –

- version (str): API Version. Optional. Defaults to the latest.
- verbosity (str): *normal*, *quiet* or *full*. Defaults to *normal*.
- content_negotiation (str): *application/json* or *application/x-yaml*. Defaults to *application/json*.
- query_params (dict): Optional. See documentation.
- iterator (bool): If sets to *True* an iterator will be returned.

Returns
response (requests.models.Response).

`get_datasets`(*dataset_id: int*, ***kwargs*)

Get datasets.

Parameters

- **dataset_id** (*int*) –
- **kwargs** –
  - version (str): API Version. Optional. Defaults to the latest.
  - verbosity (str): *normal*, *quiet* or *full*. Defaults to *normal*.
  - content_negotiation (str): *application/json* or *application/x-yaml*. Defaults to *application/json*.
  - query_params (dict): Optional. See documentation.
  - iterator (bool): If sets to *True* an iterator will be returned.

Returns
response (requests.models.Response).

`get_datasets_feedback`(*dataset_id: int*, ***kwargs*)

Get dataset feedback.

Parameters

- **dataset_id** (*int*) –
- **kwargs** –
  - version (str): API Version. Optional. Defaults to the latest.
  - verbosity (str): *normal*, *quiet* or *full*. Defaults to *normal*.
  - content_negotiation (str): *application/json* or *application/x-yaml*. Defaults to *application/json*.
  - query_params (dict): Optional. See documentation.
  - iterator (bool): If sets to *True* an iterator will be returned.

Returns
response (requests.models.Response).

get_datasets_fields(*dataset_id: int*, ***kwargs*)

Get dataset fields.

Parameters

- **dataset_id** (*int*) –
- **kwargs** –
  - version (str): API Version. Optional. Defaults to the latest.
  - verbosity (str): *normal*, *quiet* or *full*. Defaults to *normal*.
  - content_negotiation (str): *application/json* or *application/x-yaml*. Defaults to *application/json*.
  - query_params (dict): Optional. See documentation.
  - iterator (bool): If sets to *True* an iterator will be returned.

Returns

response (requests.models.Response).

get_deployments(*deployment_id: Optional[str] = None*, ***kwargs*)

Get deployments.

Parameters

- **deployment_id** (*str*) –
- **kwargs** –
  - version (str): API Version. Optional. Defaults to the latest.
  - verbosity (str): *normal*, *quiet* or *full*. Defaults to *normal*.
  - content_negotiation (str): *application/json* or *application/x-yaml*. Defaults to *application/json*.
  - query_params (dict): Optional. See documentation.
  - iterator (bool): If sets to *True* an iterator will be returned.

Returns

response (requests.models.Response).

get_reports(*report_id: int*, ***kwargs*)

Get reports.

Parameters

- **report_id** (*int*) – Optional.
- **kwargs** –
  - version (str): API Version. Optional. Defaults to the latest.
  - verbosity (str): *normal*, *quiet* or *full*. Defaults to *normal*.
  - content_negotiation (str): *application/json* or *application/x-yaml*. Defaults to *application/json*.
  - query_params (dict): Optional. See documentation.
  - iterator (bool): If sets to *True* an iterator will be returned.

Returns

response (requests.models.Response).

get_reports_feedback(*report_id: int*, ***kwargs*)

Get reports feedback.

Parameters

- **report_id** (*int*) –
- **kwargs** –
  - version (str): API Version. Optional. Defaults to the latest.
  - verbosity (str): *normal*, *quiet* or *full*. Defaults to *normal*.
  - content_negotiation (str): *application/json* or *application/x-yaml*. Defaults to *application/json*.
  - query_params (dict): Optional. See documentation.
  - iterator (bool): If sets to *True* an iterator will be returned.

Returns

response (requests.models.Response).

get_reports_fields(*report_id: int*, ***kwargs*)

Get reports fields.

Parameters

- **report_id** (*int*) –
- **kwargs** –
  - version (str): API Version. Optional. Defaults to the latest.
  - verbosity (str): *normal*, *quiet* or *full*. Defaults to *normal*.
  - content_negotiation (str): *application/json* or *application/x-yaml*. Defaults to *application/json*.
  - query_params (dict): Optional. See documentation.
  - iterator (bool): If sets to *True* an iterator will be returned.

Returns

response (requests.models.Response).

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

Test the API's availability.

It returns 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 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*, *query_params: Optional[dict] = None*, *version: Optional[str] = None*, *verbosity: str = 'normal'*, *content_negotiation: str = 'application/json'*) → Response

Generic method to send requests to our API.

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

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

Parameters

- **endpoint** (*str*) – API endpoint.
- **version** (*str*) – API Version. Optional. Defaults to the latest.
- **verbosity** (*str*) – *normal*, *quiet* or *full*. Defaults to *normal*.
- **content_negotiation** (*str*) – *application/json* or *application/x-yaml*. Defaults to *application/json*.
- **body** (*dict*) – Optional.
- **query_params** (*dict*) – Optional.

Returns

response (requests.models.Response).

Examples

```
>>> from mopinion 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
>>>
>>> with MopinionClient(public_key=PUBLICKEY, private_key=PRIVATEKEY) as client:
...     response = client.request("/account")
...     assert response.json()["_meta"]["code"] == 200
...     response = client.request(endpoint="/deployments")
...     assert response.json()["_meta"]["code"] == 200
>>>
>>> mopinion_client = MopinionClient(public_key=PUBLICKEY, private_
↪key=PRIVATEKEY)
>>> with mopinion_client as client:
...     response = client.request("/account")
...     assert response.json()["_meta"]["code"] == 200
...     response = client.request(endpoint="/deployments")
...     assert response.json()["_meta"]["code"] == 200
```

resource(*resource_name: str*, *resource_id: Optional[Union[str, int]] = None*, *sub_resource_name: Optional[str] = None*, *query_params: Optional[dict] = None*, *version: Optional[str] = None*, *verbosity: str = 'normal'*, *content_negotiation: str = 'application/json'*, *iterator: bool = False*) → Union[Response, 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*) – Optional.
- **sub_resource_name** (*str*) – Optional.
- **version** (*str*) – API Version. Optional. Defaults to the latest.
- **verbosity** (*str*) – *normal*, *quiet* or *full*. Defaults to *normal*.
- **content_negotiation** (*str*) – *application/json* or *application/x-yaml*. Defaults to *application/json*.
- **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

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 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
>>>
>>> with MopinionClient(public_key=PUBLICKEY, private_key=PRIVATEKEY) as client:
...     response = client.resource("account")
...     assert response.json()["_meta"]["code"] == 200
...     response = client.resource(resource_name=client.RESOURCE_ACCOUNT)
...     assert response.json()["_meta"]["code"] == 200
```

When working with the API there is a limit of elements retrieved. The `limit` parameters default 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 on the current page before retrieving the next page.

Examples

```
>>> from mopinion 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 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
```

1.4 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 the 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`

1.4.1 General

API Docs for General.

After installation, open a python terminal and set the `public_key`, and `private_key`, you can set them as environment vars.

```
>>> from mopinion import MopinionClient
>>> PUBLIC_KEY = os.environ.get("YOUR_PUBLIC_KEY")
>>> PRIVATE_KEY = os.environ.get("YOUR_PRIVATE_KEY")
>>> SIGNATURE_TOKEN = os.environ.get("YOUR_SIGNATURE_TOKEN")
```

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

```
>>> 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"
```

We can use it as a context Manager as well.

```
>>> with MopinionClient(public_key=PUBLIC_KEY, private_key=PRIVATE_KEY) as client:
...     assert client.is_available()
...     r = client.is_available(verbose=True)
...     assert r["code"] == 200 and r["response"] == "pong" and r["version"] == "2.0.0"
```

1.4.2 Examples with `mopinion.MopinionClient.resource`

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

Resource Account

API Docs for Account.

Get your account.

```
>>> response = client.resource(resource_name="account")
>>> assert response.json()["_meta"]["code"] == 200
>>> 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="application/x-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="quiet")
>>> assert "_meta" not in response.json()
```

Resource Deployments

API Docs for Deployments.

Getting deployments.

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

Getting a specific deployment.

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

Resource Datasets

API Docs for Datasets.

Getting a dataset.

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

Get fields for a dataset.

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

Resource Fields

API Docs for 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

API Docs for Feedback.

Note: There are three query parameters available for this resource.

- *limit* (int <= 100) Maximum number of results in response/
- *page* (int) Return result page.
- *filter* (string) Filter feedback results. Click here for more info about filters.

Get feedback from a dataset.

```
>>> params = {"page": 1}
>>> response = client.resource("datasets", 1234, "feedback", query_params=params)
>>> assert response.json()["_meta"]["code"] == 200
```

Get feedback for a report.

```
>>> params = {"limit": 50, "filter[ces]": "3"}
>>> response = client.resource("reports", 1234, "feedback", query_params=params)
>>> assert response.json()["_meta"]["code"] == 200
```

Resource Reports

API Docs for Reports.

Get some basic info on a report.

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

1.4.3 Examples with `mopinion.MopinionClient.request`

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

Resource Account

API Docs for 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="application/x-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="quiet")
>>> assert "_meta" not in response.json()
```

Resource Deployments

API Docs for Deployments.

Getting deployments.

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

Getting a specific deployment.

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

Resource Datasets

API Docs for Datasets.

Getting a dataset.

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

Get fields for a dataset.

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

Resource Fields

API Docs for 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

API Docs for Feedback.

Note: There are three query parameters available for this resource.

- *limit* (int <= 100) Maximum number of results in response/
- *page* (int) Return result page.
- *filter* (string) Filter feedback results. Click here for more info about filters.

Get feedback from a dataset.

```
>>> params = {"limit": 50, "filter[ces]": "3"}
>>> response = client.request("/datasets/1234/feedback", query_params=params)
>>> assert response.json()["_meta"]["code"] == 200
```

Get feedback from a report.

```
>>> params = {"page": 1}
>>> response = client.request("/reports/1234/feedback", query_params=params)
>>> assert response.json()["_meta"]["code"] == 200
```

Resource Reports

API Docs for Reports.

Get some basic info on a report.

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

1.4.4 Examples with GET methods

Resource Account

API Docs for Account.

Get your account.

```
>>> response = client.get_account()
>>> assert response.json()["_meta"]["code"] == 200
>>> 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.get_account(content_negotiation="application/x-yaml")
>>> r = yaml.safe_load(response.text)
>>> assert r["_meta"]["code"] == 200
```

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

```
>>> response = client.get_account(verbosity="quiet")
>>> assert "_meta" not in response.json()
```

Resource Deployments

API Docs for Deployments.

Getting deployments.

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

Getting a specific deployment.

```
>>> response = client.get_deployments(deployment_id="my_deployment_id")
>>> assert response.json()["_meta"]["code"] == 200
```

Resource Datasets

API Docs for Datasets.

Getting specific dataset.

```
>>> response = client.get_datasets(dataset_id=1234)
>>> assert response.json()["_meta"]["code"] == 200
```

Resource Fields

API Docs for Fields.

Get fields for a dataset.

```
>>> response = client.get_datasets_fields(dataset_id=1234)
>>> assert response.json()["_meta"]["code"] == 200
```

Get fields for a report.

```
>>> response = client.get_reports_fields(report_id=1234)
>>> assert response.json()["_meta"]["code"] == 200
```

Resource Feedback

API Docs for Feedback.

Note: There are three query parameters available for this resource.

- *limit* (int <= 100) Maximum number of results in response/
- *page* (int) Return result page.
- *filter* (string) Filter feedback results. Click here for more info about filters.

Get feedback from a dataset.

```
>>> params = {"page": 1}
>>> response = client.get_datasets_feedback(dataset_id=1234, query_params=params)
>>> assert response.json()["_meta"]["code"] == 200
```

Get feedback for a report.

```
>>> params = {"limit": 50, "filter[ces]": "3"}
>>> response = client.get_reports_feedback(report_id=1234, query_params=params)
>>> assert response.json()["_meta"]["code"] == 200
```

Resource Reports

API Docs for Reports.

Get some basic info on a report.

```
>>> response = client.get_reports(report_id=1)
>>> assert response.json()["_meta"]["code"] == 200
```

1.4.5 Examples with the iterator

When working with the API there is a limit of elements retrieved. The `limit` parameters default 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 on the current page before retrieving the next page.

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

Requesting fields for a dataset.

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

Also, for example, requesting fields for a report.

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

1.5 License

1.6 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 Contact Us.

CHAPTER

TWO

INDICES AND TABLES

- genindex
- modindex
- search

PYTHON MODULE INDEX

m

mopinion, 5

INDEX