pinata-python

Release 1.0.0

Panagiotis Efstratiou

Apr 25, 2022

Chapter 1

Base API

class `pinata_python.base.`**`Pinata`** (*PINATA_API_KEY: Optional[str] = None, PINATA_API_SECRET: Optional[str] = None, PINATA_JWT_TOKEN: Optional[str] = None, AUTH: str = 'keysecret'*)

Bases: `object`

Pinata Base class.

Attributes:

PINATA_API_KEY (str, Optional): Pinata's api_key.

PINATA_API_SECRET (str, Optional): Pinata's api_secret.

PINATA_JWT_TOKEN (str, Optional): Pinata's JWT token.

AUTH (str, Optional): Authentication method. Default to 'keysecret'. Otherwise, 'jwt'.

Example:

```python
from pinata_python.base import Pinata

# keysecret
pinata = Pinata(
                PINATA_API_KEY='<YOUR-API-KEY>',
                PINATA_API_SECRET='<YOUR-API-SECRET>'
            )

# jwt
pinata = Pinata(AUTH="jwt", PINATA_JWT_TOKEN='<YOUR-JWT>')
```

`AUTH`: str = 'keysecret'

`DATA_USAGE`: ClassVar[str] = 'data/userPinnedDataTotal'

`GENERATE_API_KEYS`: ClassVar[str] = 'users/generateApiKey'

`GLOBAL_PIN_POLICY`: ClassVar[str] = 'pinning/userPinPolicy'

`HASH_METADATA`: ClassVar[str] = 'pinning/hashMetadata'

`HASH_PIN_POLICY`: ClassVar[str] = 'pinning/hashPinPolicy'

`PINATA_API_KEY`: str = None

`PINATA_API_SECRET`: str = None

`PINATA_BASE_URL`: ClassVar[str] = 'https://api.pinata.cloud/'

`PINATA_IPFS_GATEWAY`: ClassVar[str] = 'https://gateway.pinata.cloud/ipfs/'

PINATA_JWT_TOKEN: str = None

PIN_BY_HASH: ClassVar[str] = 'pinning/pinByHash'

PIN_FILE_TO_IPFS: ClassVar[str] = 'pinning/pinFileToIPFS'

PIN_JOBS: ClassVar[str] = 'pinning/pinJobs'

PIN_JSON_TO_IPFS: ClassVar[str] = 'pinning/pinJSONToIPFS'

QUERY_PINS: ClassVar[str] = 'data/pinList'

REVOKE_API_KEY: ClassVar[str] = 'users/revokeApiKey'

UNPIN: ClassVar[str] = 'pinning/unpin/'

headers ( ) → Dict[str, str]

Get request's headers.

Raises:

AuthenticationError: If api_key or api_secret have not been provided.
AuthenticationError: If JWT token has not been provided.
AuthenticationError: No authentication way has been declared.

Returns:

Headers: The authentication headers.

set_authentication (*options: Optional[Dict[str, Any]] = None*) → bool

Set new authentication credentials.

Args:

options (Options, optional): The dictionary with keywords 'jwt_token' or ('api_key' and 'api_secret'). Defaults to None.

Raises:

SetAuthenticationError: If both authentication methods has been defined in the dictionary.

Returns:

bool: True. Otherwise, False.

Example:

```
from pinata_python.base import Pinata

pinata = Pinata(
                PINATA_API_KEY='<YOUR-API-KEY>',
                PINATA_API_SECRET='<YOUR-API-SECRET>'
            )
# OR pinata = Pinata(AUTH="jwt", PINATA_JWT_TOKEN='<YOUR-JWT>')

options = {'jwt_token': '<YOUR-NEW-JWT>'}
# OR options = {'api_key': '<YOUR-NEW-API-KEY>', 'api_secret':
'<YOUR-NEW-API-SECRET>'

response = pinata.set_authentication(options=options)
print(response)
```

Results:

```
True
```

Users API

class `pinata_python.users.`**`Users`** (*PINATA_API_KEY: Optional[str] = None, PINATA_API_SECRET: Optional[str] = None, PINATA_JWT_TOKEN: Optional[str] = None, AUTH: str = 'keysecret'*)

Bases: `pinata_python.base.Pinata`

Extended Users class. Inherits from Pinata.

Args:

Pinata (class): pinata_python.base.Pinata

`generate_keys` (*key_name: str = 'default_pinata', admin: bool = False, pin_list: bool = True, user_pinned_data_total: bool = True, hash_metadata: bool = True, hash_pin_policy: bool = True, pin_by_hash: bool = True, pin_file_to_ipfs: bool = True, pin_json_to_ipfs: bool = True, pin_jobs: bool = True, unpin: bool = True, user_pin_policy: bool = True, options: Optional[Dict[str, Any]] = None*) → Union[Dict[str, str], Dict[str, int], Dict[str, Any]]

See (https://docs.pinata.cloud/user/generate-api-key).

Args:

key_name (str, optional): The name of the key. Defaults to "default_pinata".
admin (bool, optional): Admin permissions. Defaults to False.
pin_list (bool, optional): API Endpoint Access. Defaults to True.
user_pinned_data_total (bool, optional): API Endpoint Access. Defaults to True.
hash_metadata (bool, optional): API Endpoint Access. Defaults to True.
hash_pin_policy (bool, optional): API Endpoint Access. Defaults to True.
pin_by_hash (bool, optional): API Endpoint Access. Defaults to True.
pin_file_to_ipfs (bool, optional): API Endpoint Access. Defaults to True.
pin_json_to_ipfs (bool, optional): API Endpoint Access. Defaults to True.
pin_jobs (bool, optional): API Endpoint Access. Defaults to True.
unpin (bool, optional): API Endpoint Access. Defaults to True.
user_pin_policy (bool, optional): API Endpoint Access. Defaults to True.
options (Options, optional): The dictionary with keyword max_uses. Defaults to None.

Returns:

PinataResponse: See utils.custom_typing.py file.

Example:

```python
from pinata_python.users import Users

pinata = Users(AUTH="jwt", PINATA_JWT_TOKEN='<YOUR-JWT>')

response = pinata.generate_keys()
print(response)
```

Results:

```
{
    'JWT': str,
    'pinata_api_key': str,
    'pinata_api_secret': str
}
```

`revoke_key` (*apikey: str*) → Union[Dict[str, str], Dict[str, int], Dict[str, Any]]

See (https://docs.pinata.cloud/user/revoke-api-key).

Args:
apikey (str): The key to revoke.

Returns:
PinataResponse: See utils.custom_typing.py file.

Example:

```python
from pinata_python.users import Users

pinata = Users(AUTH="jwt", PINATA_JWT_TOKEN='<YOUR-JWT>')

key = '<YOUR-KEY>'

response = pinata.revoke_key(key)
print(response)
```

Results:

```
'Revoked'
```

Chapter 3

Data API

class `pinata_python.data.`**`Data`** (*PINATA_API_KEY: Optional[str] = None, PINATA_API_SECRET: Optional[str] = None, PINATA_JWT_TOKEN: Optional[str] = None, AUTH: str = 'keysecret'*)

Bases: `pinata_python.base.Pinata`

Extended Data class. Inherits from Pinata.

Args:

Pinata (class): pinata_python.base.Pinata

`pin_list` (*options: Optional[Dict[str, Any]] = None*) → Union[Dict[str, str], Dict[str, int], Dict[str, Any]]

See (https://docs.pinata.cloud/data/query-pins).

Args:

options (Options, optional): The filters from the available filters and/or metadata querying. Defaults to None.

Returns:

PinataResponse: See utils.custom_typing.py file.

Example:

```
from pinata_python.data import Data

pinata = Data(AUTH="jwt", PINATA_JWT_TOKEN='<YOUR-JWT>')

options = {
            'pinStart': '2022-04-16T00:00:00.000Z',
            'pinEnd': '2022-04-19T00:00:00.000Z',
            'status': 'pinned',
            'pageLimit': 1,
            'nameContains': 'oleoleole'
        }

response = pinata.pin_list(options=options)
print(response) # it's empty
```

Results:

```
{
    'count': 0,
    'rows': []
}
```

`user_pinned_data_total` ( ) → Union[Dict[str, str], Dict[str, int], Dict[str, Any]]

See (https://docs.pinata.cloud/data/data-usage).

Returns:
PinataResponse: See utils.custom_typing.py file.

Example:

```python
from pinata_python.data import Data

pinata = Data(AUTH="jwt", PINATA_JWT_TOKEN='<YOUR-JWT>')

response = pinata.user_pinned_data_total()
print(response)
```

Results:

```
{
    'pin_count': int,
    'pin_size_total': str,
    'pin_size_with_replications_total': str
}
```

Chapter 4

Gateway API

class `pinata_python.gateway.`**`PublicGateway`** (*PINATA_API_KEY: Optional[str] = None, PINATA_API_SECRET: Optional[str] = None, PINATA_JWT_TOKEN: Optional[str] = None, AUTH: str = 'keysecret'*)

Bases: `pinata_python.base.Pinata`

Extended PublicGateway class. Inherits from Pinata.

Args:

Pinata (class): pinata_python.base.Pinata

`retrieve_content` (*cid: str, save_to: str*) → Union[Dict[str, str], Dict[str, int], Dict[str, Any]]

See (https://docs.pinata.cloud/retrieving-content).

Args:

cid (str): The CID is the content identifier (also known as a hash) for the content you want to retrieve.

save_to (str): The path that you want to save your content to.

Returns:

PinataResponse: See utils.custom_typing.py file.

Example:

```
from pinata_python.gateway import PublicGateway

pinata = PublicGateway(AUTH="jwt", PINATA_JWT_TOKEN='<YOUR-JWT>')

cid = '<YOUR-CID>'
save_to = '<YOUR-PATH>'

response = pinata.retrieve_content(cid, save_to)
print(response)
```

Results:

```
200
```

Pinning API

class pinata_python.pinning.**Pinning** (*PINATA_API_KEY: Optional[str] = None*, *PINATA_API_SECRET: Optional[str] = None*, *PINATA_JWT_TOKEN: Optional[str] = None*, *AUTH: str = 'keysecret'*)

Bases: pinata_python.base.Pinata

Extended Pinning class. Inherits from Pinata.

Args:

Pinata (class): pinata_python.base.Pinata

global_pin_policy (*new_pin_policy: Dict[str, Any]*, *migrate_previous_pins: bool = True*) → Union[Dict[str, str], Dict[str, int], Dict[str, Any]]

See (https://docs.pinata.cloud/api-pinning/global-pin-policy).

Args:

new_pin_policy (Dict[str, Any]): The new pin policy that you want to apply.
migrate_previous_pins (bool, optional): Migrating your previous pins means that all of your existing content on Pinata will be replicated to match your new pin policy. Defaults to True.

Returns:

PinataResponse: See utils.custom_typing.py file.

Example:

```
from pinata_python.pinning import Pinning

pinata = Pinning(AUTH="jwt", PINATA_JWT_TOKEN='<YOUR-JWT>')

new_pin_policy = {
                    'regions': [
                        {
                            'id': 'FRA1',
                            'desiredReplicationCount': 1
                        }
                    ]
                }
response = pinata.global_pin_policy(
                    new_pin_policy,
                    migrate_previous_pins=True
                )
print(response)
```

Results:

```
200
```

hash_metadata (*ipfs_pin_hash: str, options: Optional[Dict[str, Any]] = None*) → Union[Dict[str, str], Dict[str, int], Dict[str, Any]]

See (https://docs.pinata.cloud/api-pinning/hash-metadata).

Args:

ipfs_pin_hash (str): The hash of the content you with to change the pin policy for.
options (Options, optional): The dictionary with keywords 'name' and/or 'keyvalues'. Defaults to None.

Returns:

PinataResponse: See utils.custom_typing.py file.

Example:

```
from pinata_python.pinning import Pinning

pinata = Pinning(AUTH="jwt", PINATA_JWT_TOKEN='<YOUR-JWT>')

cid = '<YOUR-CID>'
options = {
             'name': 'alice',
             'keyvalues': {
                    'power': 100,
                    'stamina': 'high',
                    'job': 'queen'
            }
        }

response = pinata.hash_metadata(cid, options)
print(response)
```

Results:

```
200
```

hash_pin_policy (*ipfs_pin_hash: str, new_pin_policy: Dict[str, Any]*) → Union[Dict[str, str], Dict[str, int], Dict[str, Any]]

See (https://docs.pinata.cloud/api-pinning/hash-pin-policy).

Args:

ipfs_pin_hash (str): The hash of the content you with to change the pin policy for.
new_pin_policy (Dict[str, Any]): The new pin policy that you want to apply.

Returns:

PinataResponse: See utils.custom_typing.py file.

Example:

```
from pinata_python.pinning import Pinning

pinata = Pinning(AUTH="jwt", PINATA_JWT_TOKEN='<YOUR-JWT>')

cid = '<YOUR-CID>'
new_pin_policy = {
                    'regions': [
                        {
                            'id': 'FRA1',
                            'desiredReplicationCount': 1
                        }
                    ]
                }

response = pinata.hash_pin_policy(cid, new_pin_policy)
print(response)
```

Results:

```
200
```

pin_by_hash (*hash_to_pin: str*, *options: Optional[Dict[str, Any]] = None*) → Union[Dict[str, str], Dict[str, int], Dict[str, Any]]

See (https://docs.pinata.cloud/api-pinning/pin-by-hash).

Args:

hash_to_pin (str): The hash string that you want to pin.

options (Options, optional): The dictionary with keywords 'pinataOptions' and/or 'pinataMetadata'. Defaults to None.

Returns:

PinataResponse: See utils.custom_typing.py file.

Example:

```
from pinata_python.pinning import Pinning

pinata = Pinning(AUTH="jwt", PINATA_JWT_TOKEN='<YOUR-JWT>')

options = {
            'pinataMetadata': {
                'name': 'alice',
                'keyvalues': {
                    'power': 100,
                    'stamina': 'high',
                    'job': 'queen'
                }
            }
        }
hashstring = '<YOUR-HASHSTRING>'

response = pinata.pin_by_hash(hashstring, options)
print(response)
```

Results:

```
{
    'id': str,
    'IpfsHash': str,
    'PinSize': int,
    'Timestamp': str, # ISO 8601 format
}
```

pin_file_to_ipfs (*filepath: str*, *extension: str = ''*) → Union[Dict[str, str], Dict[str, int], Dict[str, Any]]

See (https://docs.pinata.cloud/api-pinning/pin-file).

Args:

filepath (str): The path of the desired file.

extension (str): The desired file extension, i.e. '.png'. Defaults to '', which means all files.

Returns:

PinataResponse: See utils.custom_typing.py file.

Example:

```
from pinata_python.pinning import Pinning

pinata = Pinning(AUTH="jwt", PINATA_JWT_TOKEN='<YOUR-JWT>')

filepath = '<YOUR-FILEPATH>'
```

```
response = pinata.pin_file_to_ipfs(filepath)
print(response)
```

Results:

```
{
     'IpfsHash': str,
     'PinSize': int,
     'Timestamp': str, # ISO 8601 format
}
```

pin_jobs (*options: Optional[Dict[str, Any]] = None*) → Union[Dict[str, str], Dict[str, int], Dict[str, Any]]

See (https://docs.pinata.cloud/api-pinning/pin-jobs).

Args:

options (Options, optional): The dictionary with the available filters as keywords. Defaults to None.

Returns:

PinataResponse: See utils.custom_typing.py file.

Example:

```
from pinata_python.pinning import Pinning

pinata = Pinning(AUTH="jwt", PINATA_JWT_TOKEN='<YOUR-JWT>')

options = {
            'sort': 'ASC',
            'status': 'prechecking',
            'limit': 3
        }
response = pinata.pin_jobs(options)
print(response) # it's empty
```

Results:

```
{
    'count': 0,
    'rows': []
}
```

pin_json_to_ipfs (*json_struct: Dict[str, Any], options: Optional[Dict[str, Any]] = None*) → Union[Dict[str, str], Dict[str, int], Dict[str, Any]]

See (https://docs.pinata.cloud/api-pinning/pin-json).

Args:

json_struct (Dict[str, Any]): The json file.

options (Options, optional): The dictionary with keywords 'pinataOptions' and/or 'pinataMetadata'. Defaults to None.

Returns:

PinataResponse: See utils.custom_typing.py file.

Example:

```
from pinata_python.pinning import Pinning

pinata = Pinning(AUTH="jwt", PINATA_JWT_TOKEN='<YOUR-JWT>')

player = {'name': 'goku', 'surname': 'son goku'}
options = {
    'pinataMetadata': {
        'name': 'player1',
        'keyvalues': {
            'power': 10,
            'stamina': 'high',
            'skill': 'jumping'
        }
    }
}

response = pinata.pin_json_to_ipfs(player, options)
print(response)
```

Results:

```
{
    'IpfsHash': str,
    'PinSize': int,
    'Timestamp': str, # ISO 8601 format
}
```

unpin (*hash_string: str*) → Union[Dict[str, str], Dict[str, int], Dict[str, Any]]

See (https://docs.pinata.cloud/api-pinning/unpin).

Args:

hash_string (str): The hash string that you want to unpin.

Returns:

PinataResponse: See utils.custom_typing.py file.

Example:

```
from pinata_python.pinning import Pinning

pinata = Pinning(AUTH="jwt", PINATA_JWT_TOKEN='<YOUR-JWT>')

cid = '<YOUR-CID>'

response = pinata.unpin(cid)
print(response)
```

Results:

```
200
```

Chapter 6

Helpers module

6.1 exceptions.exceptions submodule

exception `pinata_python.exceptions.exceptions.`**`AuthenticationError`** (*msg='Authentication is not provided.'*)

Bases: `Exception`

Authentication exception.

exception `pinata_python.exceptions.exceptions.`**`SetAuthenticationError`** (*msg=''*)

Bases: `Exception`

Set authentication exception.

- genindex
- modindex
- search

Python Module Index