moneybird-python Documentation

Release 0.1.3

Jan-Jelle Kester

Jun 25, 2017

Contents

The `moneybird-python` package provides a clean interface to the MoneyBird API.

The package is aimed at a low level abstraction to make working with the API easier but keep maximum flexibility and low overhead.

`moneybird-python` uses the excellent `requests` package.

CHAPTER 1

Table of contents

Installation

At this moment the package can only be installed from source. The source files can be downloaded from GitHub.

Authentication

Two authentication methods are supported:

- Token authentication (*TokenAuthentication*)
- OAuth authentication (*OAuthAuthentication*)

Token authentication

Token authentication is the simplest form of authentication. In the MoneyBird application a token can be generated, which you can pass to your *TokenAuthentication* instance.

Token authentication is useful when your application only accesses a single MoneyBird administration. Token authentication is not recommended when you want to access multiple administrations, especially not when you access administrations belonging to users of your application. Please use OAuth authentication instead.

```
from moneybird import TokenAuthentication

auth = TokenAuthentication('my_moneybird_token')
```

Warning: Never include your token in your source code since this token can be (mis)used to access your MoneyBird account!

I recommend to pass your token to your application using a local configuration file, or even better, an environment variable.

OAuth authentication

OAuth authentication can be used when your application accesses multiple administrations. Using OAuth, the user authorizes your application using a secure process. OAuth requires interaction with the user to work.

This documentation assumes that the reader has a sufficient knowledge of the OAuth technology and the processes related to it.

Details about the MoneyBird OAuth implementation can be found here.

Prerequisites

For OAuth authentication your application has to be registered in MoneyBird. MoneyBird will provide you with a *client id* and a *client secret*. Both these values are required by the OAuth authentication implementation.

> **Warning:** Never include your client id and/or client secret in your source code!

The OAuth authentication can be set up as follows:

```
from moneybird import OAuthAuthentication

auth = OAuthAuthentication(
    redirect_url='https://yoursite.example.com/oauth/callback/',
    client_id='your_client_id',
    client_secret='your_client_secret',
)
```

Requesting authorization

Before you can do any API calls, an access token needs to be obtained. This can be done by redirecting the user to the authorize url. This url can be obtained using *OAuthAuthentication.authorize_url()*.

The response from MoneyBird can be processed and exchanged for an access token using *OAuthAuthentication.obtain_token()*.

Authenticating a user

When an access token has been obtained this token can be used to perform API calls. The *OAuthAuthentication* instance can be (re)used, or the obtained token can be used with a new *TokenAuthentication* instance.

```
from moneybird import MoneyBird, OAuthAuthentication

auth = OAuthAuthentication(
    redirect_url='https://yoursite.example.com/oauth/callback/',
    client_id='your_client_id',
    client_secret='your_client_secret',
)
auth.obtain_token('https://yoursite.example.com/oauth/callback/?code=any&state=random_
↪string', 'random_string')
```

```
moneybird = MoneyBird(auth)
```

```
from moneybird import MoneyBird, OAuthAuthentication

auth = OAuthAuthentication(
    redirect_url='https://yoursite.example.com/oauth/callback/',
    client_id='your_client_id',
    client_secret='your_client_secret',
)
access_token = auth.obtain_token('https://yoursite.example.com/oauth/callback/?
↪code=any&state=random_string', 'random_string')

moneybird = MoneyBird(TokenAuthentication(access_token))
```

The access token can be stored for later use. At the moment of writing MoneyBird access tokens do not expire. However, a user might remove the authorization for the token, making the API inaccessible using the token.

For convenience, `OAuthAuthentication.__init__()` also acceps an `auth_token` parameter. This enables you to always use an *`OAuthAuthentication`* instance regardless of whether you already have a token or not.

Internal API

class moneybird.authentication.**Authentication**

Bases: object

Base class for authentication implementations.

get_session() → requests.sessions.Session

Creates a new session with the authentication settings applied.

Returns The new session

is_ready() → bool

Checks whether authentication can be performed. A negative result means that it is certain that a request will not authenticate.

Returns Whether the authentication is ready to be used

class moneybird.authentication.**OAuthAuthentication**(*redirect_url: str*, *client_id: str*, *client_secret: str*, *auth_token: str = ''*)

Bases: *moneybird.authentication.Authentication*

OAuth authentication for the MoneyBird API.

This is a wrapper around TokenAuthentication since token authentication is used after the OAuth process has been performed. This authentication method cannot be used directly, some work is required since the user has to perform a number of actions before a token can be obtained.

Parameters

- **redirect_url** – The URL to redirect to after successful authorization
- **client_id** – The OAuth client id obtained from MoneyBird
- **client_secret** – The OAuth client secret obtained from MoneyBird
- **auth_token** – The optional token from an earlier authorization

exception `OAuthError`(*error_code: str*, *description: str = None*)
Bases: `Exception`

Exception for OAuth protocol errors.

`OAuthAuthentication.`**`authorize_url`**(*scope: list*, *state: str = None*) → tuple
Returns the URL to which the user can be redirected to authorize your application to access his/her account and the state which can be used for CSRF protection as a tuple.

Example:

```
>>> auth = OAuthAuthentication('https://example.com/oauth/moneybird/',
↪'your_id', 'your_secret')
>>> auth.authorize_url()
('https://moneybird.com/oauth/authorize?client_id=your_id&redirect_
↪uri=https%3A%2F%2Fexample.com%2Flogin%2F
moneybird&state=random_string', 'random_string')
```

Parameters

- **`scope`** – The requested scope
- **`state`** – Optional state, when omitted a random value is generated

Returns 2-tuple containing the URL to redirect the user to and the randomly generated state

`OAuthAuthentication.`**`obtain_token`**(*redirect_url: str*, *state: str*) → str
Exchange the code obtained using *authorize_url* for an authorization token.

Example:

```
>>> auth = OAuthAuthentication('https://example.com/oauth/moneybird/',
↪'your_id', 'your_secret')
>>> auth.obtain_token('https://example.com/oauth/moneybird/?code=any&
↪state=random_string', 'random_string')
'token_for_auth'
>>> auth.is_ready()
True
```

Parameters

- **`redirect_url`** – The full URL the user was redirected to
- **`state`** – The state used in the authorize url

Returns The authorization token

class `moneybird.authentication.`**`TokenAuthentication`**(*auth_token: str = ''*)
Bases: *`moneybird.authentication.Authentication`*

Token authentication for the MoneyBird API.

Parameters **`auth_token`** – The authentication token to use.

`set_token`(*auth_token: str*)
Sets the authentication token.

Parameters **`auth_token`** – The authentication token to use.

Accessing the API

The *MoneyBird* class provides a low level of abstraction for communicating with the MoneyBird API. This means that the library only provides ways to minimize repetitive work, but does not contain an internal data representation of the MoneyBird data.

API docs

Knowledge of the MoneyBird API is expected and required for working with this library. API documentation can be fount at http://developer.moneybird.com/.

Set up

Before querying the API, it has to be set up properly. Every instance of the API requires a valid form of authentication:

```
from moneybird import MoneyBird, TokenAuthentication

moneybird = MoneyBird(TokenAuthentication('token'))
```

See usage/authentication for details on authentication.

Queries

For queries, there are four methods, one for each type:

- get: *MoneyBird.get()*
- post: *MoneyBird.post()*
- patch: *MoneyBird.patch()*
- delete: *MoneyBird.delete()*

Method signatures

The method signatures of these methods are pretty similar.

For all methods, the first argument is the resource url, as given by the MoneyBird API documentation, excluding the domain, version, format, etc. So for the list of contacts this will be `contacts`, for a specific contact this will be `contacts/%(id)s`.

For the post and patch methods, which send data, the second argument is the data that is to be sent. The data should be regular Python objects ready for JSON serializing. Dictionaries and lists are commonly used. The data should be formatted according to the appropriate format for the resource.

The last argument, with the keyword `administration_id`, should contain the administration id when this is required for the used resource.

The methods always return the response from the API or throw an exception. The response will be a Python object built from the JSON response.

The methods are described in detail below.

Example

```
from moneybird import MoneyBird, TokenAuthentication

# API client
moneybird = MoneyBird(TokenAuthentication('token'))

# Get an administration id
administrations = moneybird.get('administrations')

# Get all contacts for all administrations
for administration in administrations:
    id = administration['id']
    contacts = moneybird.get('contacts', administration_id=id)

    # Print invoices per contact
    for contact in contacts:
        print(contact['company_name'])

        for invoice in moneybird.get('sales_invoices?filter=contact_id:%s' % contact[
↪'id'], administration_id=id)
            print('  ', invoice['invoice_id'])
```

Internal API

class moneybird.api.**MoneyBird**(*authentication: moneybird.authentication.Authentication*)

Bases: object

Client for the MoneyBird API.

Parameters **authentication** – The authentication method to use

exception **APIError**(*response: requests.models.Response*, *description: str = None*)

Bases: Exception

Exception for cases where communication with the API went wrong.

This exception is specialized into a number of exceptions with the exact same properties.

request

Short string representation of the request (method and URL).

response

JSON decoded data of the response.

status_code

HTTP status code of the request.

MoneyBird.**delete**(*resource_path: str*, *administration_id: int = None*)

Performs a DELETE request to the endpoint identified by the resource path. DELETE requests are usually used to (permanently) delete existing data. USE THIS METHOD WITH CAUTION.

From a client perspective, DELETE requests behave similarly to GET requests.

Parameters

- **resource_path** – The resource path
- **administration_id** – The administration id (optional, depending on the resource path)

Returns The decoded JSON response for the request

MoneyBird.**get**(*resource_path: str*, *administration_id: int = None*)

Performs a GET request to the endpoint identified by the resource path.

Example:

```
>>> from moneybird import MoneyBird, TokenAuthentication
>>> moneybird = MoneyBird(TokenAuthentication('access_token'))
>>> moneybird.get('administrations')
[{'id': 123, 'name': 'Parkietje B.V.', 'language': 'nl', ...
>>> moneybird.get('contacts/synchronization', 123)
[{'id': '143273868766741508', 'version': 1450856630}, ...
```

Parameters

- **resource_path** – The resource path
- **administration_id** – The administration id (optional, depending on the resource path)

Returns The decoded JSON response for the request

MoneyBird.**patch**(*resource_path: str*, *data: dict*, *administration_id: int = None*)

Performs a PATCH request to the endpoint identified by the resource path. PATCH requests are usually used to change existing data.

From a client perspective, PATCH requests behave similarly to POST requests.

Parameters

- **resource_path** – The resource path
- **data** – The data to send to the server
- **administration_id** – The administration id (optional, depending on the resource path)

Returns The decoded JSON response for the request

MoneyBird.**post**(*resource_path: str*, *data: dict*, *administration_id: int = None*)

Performs a POST request to the endpoint identified by the resource path. POST requests are usually used to add new data.

Example:

```
>>> from moneybird import MoneyBird, TokenAuthentication
>>> moneybird = MoneyBird(TokenAuthentication('access_token'))
>>> data = {'url': 'http://www.mocky.io/v2/5185415ba171ea3a00704eed'}
>>> moneybird.post('webhooks', data, 123)
{'id': '143274315994891267', 'url': 'http://www.mocky.io/v2/
↪5185415ba171ea3a00704eed', ...
```

Parameters

- **resource_path** – The resource path
- **data** – The data to send to the server
- **administration_id** – The administration id (optional, depending on the resource path)

Returns The decoded JSON response for the request

MoneyBird.**renew_session**()

Clears all session data and starts a new session using the same settings as before.

This method can be used to clear session data, e.g., cookies. Future requests will use a new session initiated with the same settings and authentication method.

CHAPTER 2

Contributing

The source code can be found (and forked) on GitHub. This is also the place where issues can be reported.

GitHub repository: https://github.com/jjkester/django-auditlog

Python Module Index

m

Index