Bexio API Python Client Documentation

Release 0.1.4

Özer Sahin

Mar 22, 2023

Contents

Contents:

CHAPTER 1

Bexio API Python Client

Bexio API Python Client. It works as a standalone lib in python, but needs additional work to get it started. There is a Django integration, that works out of the box, once you set it up properly as described later. If you wish to use it with Python alone, you can take a look at the Django integration. It shows how you can use it with python alone.

1.1 Features

- API connection to your Bexio instance
- Django integration
- Creation & Auto-renewal of access token

1.2 Documentation

The full documentation is at https://bexio-api-python-client.readthedocs.io.

1.3 Quickstart

Install Bexio API Python Client:

```
pip install bexio-api-python-client
```

If you are using anything other than Django with this module, you need to write some more code to authenticate and use the API. Check out the Django parts of the module to see how it can be done.

1.4 Django Integration

You can use the API directly with Django. There are URLs, Views and other helpers to integrate the API into your Django project.

Add it to your `INSTALLED_APPS`:

```
INSTALLED_APPS = (
    ...
    'bexiopy.apps.BexiopyConfig',
    ...
)
```

Add Bexio API Python Client's URL patterns to your main `urls.py`:

```
from bexiopy import urls as bexiopy_urls


urlpatterns = [
    ...
    url(r'bexiopy/', include('bexiopy.urls', namespace='bexiopy')),
    ...
]
```

Start your server and visit the homepage (e.g. `localhost:8000`). You should see a navigation bar at the top where you can click on "Authenticate" to create your token. Afterwards, you should see some data on the front page, if you have any data in your Bexio instance.

1.5 Docker

If you use docker, you can configure a volume to store the token:

Example:

```
# docker-compose.yml
volumes:
    bexio_token: {}

services:
    app:
        volumes:
            - bexio_token:/app/secrets

# app/Dockerfile
...
RUN mkdir /app/secrets
...
```

1.6 Basic Usage

See docs for basic usage: https://bexio-api-python-client.readthedocs.io/en/latest/bexiopy.html#bexiopy.api.Client

You can also use helper functions, so you don't need to create the `call` every time (see docs). These helper functions are added over time, but you can use the API fully, with the *call* function. The helper methods only offer nicer ways to query the API (e.g. `Bexiopy().contacts.get(2)`). The progress of the helper functions are documented below under "Progress".

1.7 Settings

Configure the minimum `settings.py` (check docs for all options):

```
BEXIO_CLIENT_SECRET = 'my_secret'
BEXIO_CLIENT_ID = 'my_id'
BEXIO_APPLICATION_SCOPES = ['my_scope_1', 'my_scope_2']
BEXIO_APPLICATION_REDIRECTION_URL = 'https://example.com'
```

Start the server and go to `/bexiopy/auth/` and authenticate with Bexio.

i18n URLs

If you have internationalized URLs, then make sure you place the Bexiopy url outside the internationalized ones, so it can be called without any language code (`/bexiopy/auth/` instead of `/en/bexiopy/auth/`).

1.8 Running Tests

Does the code actually work?

```
source <YOURVIRTUALENV>/bin/activate
(myenv) $ pip install tox
(myenv) $ tox
```

1.9 Credits

Tools and libs used in creating this package:

- Cookiecutter
- cookiecutter-djangopackage
- Christian Ruhstaller (bexio-api-php-client)

1.10 Progress

[ ] = Open

[~] = In Progress

[X] = Done

[-] = Not possible / Ignored

1.10.1 Contacts

Contacts

- [X] List contacts
- [X] Search contacts
- [X] Show contact
- [X] Create contact
- [X] Overwrite contact
- [X] Edit contact
- [X] Delete contact
- [ ] Bulk create contact

Salutations

- [ ] List salutations
- [ ] Search salutations
- [ ] Show salutation
- [ ] Create salutation
- [ ] Overwrite salutation
- [ ] Edit salutation
- [ ] Delete salutation

Titles

- [ ] List titles
- [ ] Search titles
- [ ] Show title
- [ ] Create title
- [ ] Overwrite title
- [ ] Edit title
- [ ] Delete title

1.10.2 Invoices

- [X] List invoices
- [X] Search invoices
- [X] Show invoice
- [X] Create invoice
- [X] Overwrite invoice

- [X] Edit invoice
- [X] Delete invoice
- [X] Show invoice pdf
- [X] Copy invoice
- [ ] Issue invoice
- [ ] Mark invoice as sent
- [ ] Send invoice
- [ ] List comments
- [ ] Search comments
- [ ] Show comment
- [ ] Create comment
- [-] List payments
- [-] Show payments
- [-] Create payments
- [-] Delete payments

1.11 Sponsorship

This project is maintained by Mathison AG | Mobile & Web Development.

CHAPTER 2

Package

2.1 API

2.2 Settings

bexiopy.settings.**get_setting**(*name*)

Receive a name and try to return the corresponding setting.

Parameters **name** (*str*) – name of setting

Returns value of requested setting

Return type mixed

Django

The settings defined in the root settings of your django project have priority over defaults.

Python

If you use the API without Django, you can define the settings in this file. Just add the following to the top of the file with your settings:

```
settings = {
    'BEXIO_AUTH_URL': '...',
    'BEXIO_CLIENT_ID': '...',
    'BEXIO_CLIENT_SECRET': '...',
    ...
}
```

2.3 Django Views

CHAPTER 3

Resources

3.1 Base

class bexiopy.resources.base.**BaseClientResource**
Bases: object

Base client resource that instantiates the Client.

client

class bexiopy.resources.base.**BaseResource**(**args*, ***kwargs*)
Bases: *bexiopy.resources.base.BaseClientResource*

Base resource that's inherited by all other resources.

Inheriting classes may have additional methods, that are resource specific. Take a look at the resources to find these additional methods.

ENDPOINT
the endpoint that should be queried

Type str

ENDPOINT_SEARCH
the serach endpoint that should be queried

Type str

all()
Get all objects of given endpoint.

Returns List of all objects from requested endpoint.

Return type list

create(*data*)
Add new object.

Parameters **data** (dict) – Dictionary object with appropriate data.

Returns Object that has been created.

Return type dict

delete(*pk*)
Delete object.

Parameters **pk** (*str*) – Bexio id of object.

Returns Response dictionary of operation.

Return type dict

get(*pk*)
Get specific object.

Parameters **pk** (*str*) – Bexio id of object.

Returns Object that has been pulled.

Return type dict

get_or_create(*pk=None*, *data={}*)
Return object if exists, else create object first.

Parameters

- **pk** (*str*) – Bexio id of object.
- **data** (*dict*) – Data for possible object creation.

Returns Object that has been pulled.

Return type dict

overwrite(*pk*, *data*)
Add new contact

Parameters

- **pk** (*str*) – Bexio id of object.
- **data** (*dict*) – Data that should be overwritten.

Returns Object that has been overwritten.

Return type dict

search(*params=[]*)
Search for specific object and return response.

Parameters **params** (list of dict, optional) – Parameters to narrow down the search.

Returns List of results from request.

Return type list

update(*pk*, *data*)
Update existing object.

Parameters

- **pk** (*str*) – Bexio id of object.
- **data** (*dict*) – Data that should be updated.

Returns Object that has been updated.

Return type dict

update_or_create(*pk*, *data*)
Update object if exists, else create object.

Parameters

- **pk** (*str*) – Bexio id of object.
- **data** (*dict*) – data for object update/creation.

Returns Object that has been pulled.

Return type dict

3.2 Contacts

class bexiopy.resources.contacts.**ContactsResource**(**args*, ***kwargs*)
Bases: *bexiopy.resources.base.BaseResource*

Resource to query the contacts endpoint.

Endpoint Docs:

https://docs.bexio.com/ressources/contact/

Examples:

```
bexio = Bexiopy()

# get all contacts
contacts = bexio.contacts.all()

# create an invoice
contact = bexio.invoices.create(params={'attr1': 'val1', ...)

# search a contact
contact = bexio.contacts.search(params={'param1': 'some value'})

# get one specific contact with id 2
contact = bexio.contacts.get(pk=2)
```

ENDPOINT = 'contact'

ENDPOINT_SEARCH = 'contact/search'

create_contact_relation(*id*, *sub_id*, *desc=''*)
Create a contact relationship

Parameters

- **id** (*int*) – Bexio id of parent contact
- **sub_id** (*int*) – Bexio id of child contact

Returns Dict of saved data

Return type dict

get_relations()
Get relations from contacts

Returns List of contact relations

Return type list

3.3 General

class bexiopy.resources.general.**GeneralResource**
Bases: *bexiopy.resources.base.BaseClientResource*

Resource to get general information about your Bexio instance.

get_salutations()
Get available salutations

Returns mixed

get_titles()
Get available titles

Returns mixed

3.4 Invoices

class bexiopy.resources.invoices.**InvoicesResource**(**args*, ***kwargs*)
Bases: *bexiopy.resources.base.BaseResource*

Resource to query the contacts endpoint.

Endpoint Docs:

https://docs.bexio.com/ressources/kb_invoice/

Examples:

```
from bexiopy.api import *

api = Bexiopy()

# create invoice
invoice = api.invoices.create(invoice_data)

# get an invoice
invoice = api.invoices.get(23)

# update invoice
invoice = api.invoices.update(23,
    {
        'user_id': 1,
        'contact_id': 2,
        'header': 'New Header'
    }
)

# delete invoice
invoice = api.invoices.delete(23)

# search for invoices
api.invoices.search(
    [
        {'field': 'user_id', 'value': 1},
        {'field': 'contact_id', 'value': 2}
```

(continues on next page)

(continued from previous page)

```
    ]
)
```

ENDPOINT = 'kb_invoice'

ENDPOINT_SEARCH = 'kb_invoice/search'

copy(*pk*, *data*)

Return the PDF version of a given invoice.

Parameters **pk** (*str*) – Bexio id of invoice

Returns Invoice that was copied

Return type dict

show_pdf(*pk*)

Return the PDF version of a given invoice. Needs to be further processed so you can extract the file from the response.

Parameters **pk** (*str*) – Bexio id of invoice

Returns PDF file of invoice

Return type file

CHAPTER 4

Contributing

Contributions are welcome, and they are greatly appreciated! Every little bit helps, and credit will always be given.

You can contribute in many ways:

4.1 Types of Contributions

4.1.1 Report Bugs

Report bugs at https://github.com/oesah/bexio-api-python-client/issues.

If you are reporting a bug, please include:

- Your operating system name and version.
- Any details about your local setup that might be helpful in troubleshooting.
- Detailed steps to reproduce the bug.

4.1.2 Fix Bugs

Look through the GitHub issues for bugs. Anything tagged with “bug” is open to whoever wants to implement it.

4.1.3 Implement Features

Look through the GitHub issues for features. Anything tagged with “feature” is open to whoever wants to implement it.

4.1.4 Write Documentation

Bexio API Python Client could always use more documentation, whether as part of the official Bexio API Python Client docs, in docstrings, or even on the web in blog posts, articles, and such.

4.1.5 Submit Feedback

The best way to send feedback is to file an issue at https://github.com/oesah/bexio-api-python-client/issues.

If you are proposing a feature:

- Explain in detail how it would work.
- Keep the scope as narrow as possible, to make it easier to implement.
- Remember that this is a volunteer-driven project, and that contributions are welcome :)

4.2 Get Started!

Ready to contribute? Here's how to set up *bexio-api-python-client* for local development.

1. Fork the *bexio-api-python-client* repo on GitHub.
2. Clone your fork locally:

```
$ git clone git@github.com:your_name_here/bexio-api-python-client.git
```

3. Install your local copy into a virtualenv. Assuming you have virtualenvwrapper installed, this is how you set up your fork for local development:

```
$ mkvirtualenv bexio-api-python-client
$ cd bexio-api-python-client/
$ python setup.py develop
```

4. Create a branch for local development:

```
$ git checkout -b name-of-your-bugfix-or-feature
```

   Now you can make your changes locally.

5. When you're done making changes, check that your changes pass flake8 and the tests, including testing other Python versions with tox:

```
$ flake8 bexiopy tests
$ python setup.py test
$ tox
```

   To get flake8 and tox, just pip install them into your virtualenv.

6. Commit your changes and push your branch to GitHub:

```
$ git add .
$ git commit -m "Your detailed description of your changes."
$ git push origin name-of-your-bugfix-or-feature
```

7. Submit a pull request through the GitHub website.

4.3 Pull Request Guidelines

Before you submit a pull request, check that it meets these guidelines:

1. The pull request should include tests.
2. If the pull request adds functionality, the docs should be updated. Put your new functionality into a function with a docstring, and add the feature to the list in README.rst.
3. The pull request should work for Python 2.6, 2.7, and 3.3, and for PyPy. Check https://travis-ci.org/oesah/bexio-api-python-client/pull_requests and make sure that the tests pass for all supported Python versions.

4.4 Tips

To run a subset of tests:

```
$ python -m unittest tests.test_bexiopy
```

CHAPTER 5

Credits

5.1 Development Lead

- Özer Sahin <o.sahin@oesah.de>

5.2 Contributors

None yet. Why not be the first?

CHAPTER 6

History

6.1 0.1.5 (2017-10-23)

- added suport for Python 3.6
- changed logic, so that you can visit the index page without having a token yet

6.2 0.1.0 (2017-10-23)

- First release on PyPI.

Python Module Index

b

Index

U