Django-Mangopay Documentation

Release 0.0.1

Rebecca Meritz

May 09, 2017

Contents

Django-Mangopay is a Django wrapper for the PSP Mangopay's V2 API. It provides Django-specfic functionality around Mangopay's Python SDK. It creates a Mangopay Client via settings in your `settings.py`. It provides Django Models that allow you to persist the data that you need to send and recieve from Mangopay. These models have functions that correspond to the Mangopay's API calls. Celery tasks are also provided if you want to call these functions asynchronously.

Read extended documentation provided at Read the Docs.

Contents:

CHAPTER 1

Installation

1. Install package from PyPI.

```
pip install django-mangopay
```

2. Create a test client.
3. Add your newly created *MANGOPAY_CLIENT_ID* and *MANGOPAY_PASSPHRASE* to your django settings, as well as the *settings_base_url*.
4. To run django-mangopay in production just change the above setting to be the settings for your production server.

CHAPTER 2

Correspondance to Mangopay Rest API

This library is a wrapper of the Mangopay API as described here. Each of the API calls provided are shown with an explaination of how to use them.

Access

POST /v2/oauth/token

The client handles authentication with a token once you have set up your client as described in the installation instructions. Once confrigured it is easy to get an instance of the client.

```
import mangopay.client

client = get_mangopay_api_client()
```

Activity, research & lists

None of the API calls listed by Mangopay as falling under the category are supported. However with the client provided, it is possible to use the mangopay-sdk to call them. An example is given below.

```
from mangopay.client import get_mangopay_api_client

client = get_mangopay_api_client()
# GET /Events
client.events.Get()
```

Users

POST /users/natural

To create a natural user object just instantiate an instance of `MangoPayNaturalUser`, populate the requried fields, and call `create()` on it as shown below. You can also edit the user, just update the the values you want to change in the model and then call `update()`.

```
from django.contrib.auth.models import User
from datetime import date
from mangopay.models import MangoPayNaturalUser

user = User.objects.get(id=1)

mangopay_user = MangoPayNaturalUser()
mangopay_user.user = user
mangopay_user.country_of_residence = "SE"
mangopay_user.nationality = "US"
mangopay_user.birthday = date(1989, 10, 20)
mangopay_user.save()

mangopay_user.create()

mangopay_user.address = "Högbergsgatan 66C"
mangopay_user.save()

mangopay_user.update()
```

POST /users/legal

Creating and editing a legal user is the same as creating a natural user except Mangopay requires different fields. Instantiate the `MangoPayLegalUser`, populate the required fields, and call `create()` or `update()` on it as shown below.

```
from django.contrib.auth.models import User
from datetime import date
from mangopay.models import MangoPayLegalUser
from mangopay.constants import BUSINESS

user = User.objects.get(id=1)

mangopay_user = MangoPayLegalUser()
mangopay_user.user = user
mangopay_user.country_of_residence = "SE"
mangopay_user.nationality = "US"
mangopay_user.birthday = date(1989, 10, 20)
mangopay_user.first_name = "Rebecca"
mangopay_user.last_name = "Meritz"
mangopay_user.generic_business_email = "office@fundedbyme.com"
mangopay_user.business_name = "FundedByMe AB"
mangopay_user.type = BUSINESS
mangopay_user.save()

mangopay_user.create()

mangopay_user.headquaters_address = "Regeringsgatan 29, 111 53 Stockholm"
mangopay_user.save()

mangopay_user.update()
```

GET /users/{user_Id}

This call is not supported. Infomation about the mangopay user will already be saved on your MangoPayUserModel when you call `create` and/or `update`.

POST /KYC/Documents

To create a mangopay document for a user just instantiate a `MangoPayDocument`, save the user and type to the document, and then call `create()`. If successfully created the document's status should be updated to `CREATED` and it should be assigned a `mangopay_id`. Once you have added all the pages you wanted to the document you can ask for validation from mangopay via `ask_for_validation()`. This should change the status of the document to `VALIDATION_ASKED`.

```
from mangopay.models import MangoPayUser
from mangopay.constants import IDENTITY_PROOF

mangopay_user = MangoPayUser.objects.get(id=1)

mangopay_document = MangoPayDocument()
mangopay_document.mangopay_user = mangopay_user
mangopay_document.type = IDENTITY_PROOF
mangopay_document.save()

mangopay_document.create()

# Then add a 1+ MangoPayPages to your mangopay_document

mangopay_document.ask_for_validation()
```

POST /KYC/Documents/Pages

A document can have many pages, but needs at least one. Instantiate one `MangoPayPage` per file and call `create()` on the object to create it.

```
from mangopay.models import MangoPayPage

document = MangoPayDocument.objects.get(id=1)
file = file("tmp/file")
page = MangoPayPage(file=file, document=document)
page.save()
page.create()
```

In order for this call to work you need to decide were you want to store your files. Files can either be saved to Django's default storage by setting *MANGOPAY_PAGE_DEFAULT_STORAGE* to `True`, or you can configure your files to be stored on AWS by setting AWS storage via S3BotoStorage. `AWS_MEDIA_BUCKET_NAME` and `AWS_MEDIA_CUSTOM_DOMAIN` must be in your setting in this case.

GET /KYC/Documents/{Document_Id}

One business day after asking for validation you should be able to see if mangopay approved the document or not via `get()` which will get the updated document from mangopay. At this point it should either have the status of `VALIDATED` or `REFUSED`.

```
from mangopay.models import MangoPayDocument

document = MangoPayDocument.objects.get(id=1)
document.get()
```

POST /users/{userId}/bankaccounts/{type}

In order to complete a payout you must have registered a bank account. To do this instantiate a `MangoPayBankAccount` and add the required fields, then call `create()`. Only BIC & IBAN bank registrations are currently supported by this library.

```
from mangopay.models import MangoPayBankAccount, MangoPayUser

bank_account = MangoPayBankAccount()
bank_account.mangopay_user = MangoPayUser.objects.get(id=1)
bank_account.iban = "SE3550000000054910000003"
bank_account.bic = "53H555"
address = "Högbergsgatan 66C, 11854 Stockholm, Sweden"
```

GET /users/{userId}/bankaccounts/{id}

This call is not supported the data should already be persisted on your `MangoPayBankAccount` model.

Wallets

POST /wallets

In order to create a wallet just instantiate a `MangoPayWallet` object, add user to it, save it and call `create()`.

```
from mangopay.models MangoPayWallet, MangoPayUser

user = MangoPayUser.objects.get(id=1)
wallet = MangoPayWallet()
wallet.mangopay_user = user
wallet.save()

wallet.create(description="Sven's Wallet")
```

GET /wallets/{Wallet_Id}

`GET` is not supported directly, however you can call `balance()` on a created `MangoPayWallet` to find the amount of `Money` on the wallet.

PayIns

POST /payins/card/web

Not supported via this library or the API it is only supported by MangoPay's web interface.

POST /payins/card/direct

Once you have successfully registered a card you can create a payin from that card to a created wallet. Instantiate a `MangoPayPayIn` model, add the user, wallet, and card; then call create with the return url, the funds to be debited and optionally the fees. The payin will be created and the execution date, status, result code, id, status, and secure mode redirect url will be saved to the object.

```
from money import Money
from mangopay.models import (MangoPayPayIn, MangoPayCard, MangoPayWallet,
                             MangoPayUser)

payin = MangoPayPayIn()
payin.mangopay_user = MangoPayUser.objects.get(id=1)
payin.mangopay_wallet = MangoPayWallet.objects.get(id=1)
payin.mangopay_card = MangoPayCard.objects.get(id=1)
payin.debited_funds = Money(1001, "EUR")
payin.fees = Money(0, "EUR")
payin.create(secure_mode_return_url="https://my/secure/mode/return/url")
```

POST /payins/preauthorized/direct

Preauthorizations are not currently supported by this library. Pull requests are welcome. See *Contributing*.

GET /payins/{PayIn_Id}

Once a `MangoPayPayIn` is created it's associated status can be updated via calling `get()` on the instance.

```
from mangopay.models import MangoPayPayIn

payin = MangoPayPayIn.objects.get(id=1)
payin.get()
```

POST /cardregistration

Before a card can be used it must be registered with a user. Just instantiate a `MangoPayCardRegistration` object, add a user to it, and call `create()` with a supported currency. When you do this MangoPay's ID will be saved to the object.

```
from mangopay.models import MangoPayCardRegistration, MangoPayUser

card_registration = MangoPayCardRegistration()
card_registration.mangopay_user = MangoPayUser.objects.get(id=1)
card_registration.create("EUR")
```

GET /cardregistration/{CardRegistration_Id}

Once you have created a `MangoPayCardRegistration` object you can access the card's preregistration data by calling `get_preregistration_data()`. This data comes in the form of a dictionary with the keys: "preregistrationData", "accessKey", and "cardRegistrationURL".

```
from mangopay.models import MangoPayCardRegistration

card_registration = MangoPayCardRegistration.objects.get(id=1)
card_registration.get_preregistration_data()
```

GET /cards/{Card_Id}

After registering a card with MangoPay you should get back the card's Id. If you save that card's Id to the related `MangoPayCard` object by calling `save_mangopay_card_id()`, then later you can access the card's info by calling `request_card_info()`. Requesting the card's info will save the expiration date, alias, and active and valid state to the `MangoPayCard` object.

```
from mangopay.models import MangoPayCardRegistration

card_registration = MangoPayCardRegistration.objects.get(id=1)
card_registration.save_mangopay_card_id("123456")

card_registration.mangopay_card.request_card_info()
```

POST /preauthorization/card/direct

Preauthorizations are not currently supported by this library. Pull requests are welcome. See *Contributing*.

GET /preauthorization/{PreAuthorization_Id}

Preauthorizations are not currently supported by this library. Pull requests are welcome. See *Contributing*.

Refunds

POST /transfers/{Transfer_Id}/Refund

Transfers and refunds of those transfers are not supported by this library. Pull requests are welcome.

POST /payins/{PayIn_Id}/Refund

Currently only simple refunds are supported. That means you can only create a complete refund on a pay in, not a partial one. To create a simple refund just instantiate a `MangoPayRefund` object and add the payin you want to refund and the user; then save it and call `create_simple()`. The MangoPay's Id, the execution date, and status will be updated in the object. If the refund was successful then `create_simple` will return `True`.

```
from mangopay.models import MangoPayRefund, MangoPayPayIn

refund = MangoPayRefund()
payin = MangoPayPayIn.objects.get(id=1)
refund.payin = payin
refund.mangopay_user = payin.mangopay_user
refund.save()

refund.create_simple()
```

GET /refunds/{Refund_Id}

Getting a refund via its ID is not supported by this library. Pull requests welcome.

PayOuts

POST /payouts/bankwire

Payouts transfer money from a wallet to a user's bank account. In order for a payout to run successfully, the user's KYC requirements must be fufilled. To use it simply instantiate the `MangoPayPayOut` object add the user, the wallet you want to transfer from, and the bank account you want to transfer to, the funds to be debited, and optionally the fees to be taken; then save it and run `create()`. MangoPay's generated id, the status, and the execution date will be saved to the object.

```
from money import Money
from mangopay.models import MangoPayPayOut, MangoPayUser, MangoPayWallet, MangoPayBankAccount

payout = MangoPayPayOut()
payout.mangopay_user = MangoPayUser.objects.get(id=1)
payout.mangopay_wallet = MangoPayUser.objects.get(id=1)
payout.mangopay_bank_account = MangoPayBankAccount.objects.get(id=1)
payout.debited_funds = Money(1000, "EUR")
payout.fees = Money(10, "EUR")
payout.save()

payout.create()
```

GET /payouts/{PayOut_Id}

Getting a payout will update the status and execution date from MangoPay.

```
from mangopay.models import MangoPayPayOut

payout = MangoPayPayOut.objects.get(id=1)
payout.get():
```

Transfers

POST /payouts/bankwire

Transfer money from one mangopay wallet to another. To use it simply instantiate the `MangoPayTransfer` object add the user, the wallet you want to transfer from, and the wallet you want to transfer to, the funds to be debited; then save it and run `create()`. MangoPay's generated id, the status, and the execution date will be saved to the object.

```
from money import Money
from mangopay.models import MangoPayTransfer, MangoPayUser, MangoPayWallet

transfer = MangoPayTransfer()
transfer.mangopay_credited_wallet = MangoPayWallet.objects.get(id=1)
transfer.mangopay_debited_wallet = MangoPayWallet.objects.get(id=2)
transfer.debited_funds = Money(1000, "EUR")
```

```
transfer.save()

transfer.create()
```

GET /transfers/{Transfer_Id}

Getting a transfer will update the status and execution date from MangoPay.

```
from mangopay.models import MangoPayTransfer

transfer = MangoPayTransfer.objects.get(id=1)
transfer.get():
```

CHAPTER 3

Tasks

Celery tasks are provided. If desired you may use them to asynchcroniously call the MangoPay API.

create_mangopay_user

Takes the id of a `MangoPayUser` and creates it. See *POST /users/natural* and *post_users_legal*.

```
from django.contrib.auth.models import User
from datetime import date
from mangopay.models import MangoPayNaturalUser
from mangopay.tasks import create_mangopay_user

user = User.objects.get(id=1)

mangopay_user = MangoPayNaturalUser()
mangopay_user.user = user
mangopay_user.country_of_residence = "SE"
mangopay_user.nationality = "US"
mangopay_user.birthday = date(1989, 10, 20)
mangopay_user.save()

create_mangopay_user.delay(id=mangopay_user.id)
```

update_mangopay_user

Takes the id of a `MangoPayUser` and updates it. See *POST /users/natural* and *post_users_legal*.

create_mangopay_bank_account

Takes the id of a `MangoPayBankAccount` and creates it. See *POST /users/{userId}/bankaccounts/{type}*.

create_mangopay_document_and_pages_and_ask_for_validation

Takes the id of a `MangoPayDocument` creates the document and all the related pages and then asks for validation of the document.:ref:*UpdateDocumentsStatus* or *update_document_status'can be used to update the status. MangoPay*

says they will verify and update the status of your document the following business day. See :ref:'post_kyc_documents.

UpdateDocumentsStatus

An abstract periodic task which can be subclassed to update documents with status `VALIDATION_ASKED`.
See *GET /KYC/Documents/{Document_Id}*.

update_document_status

Takes the id of a `MangoPayDocument` and updates it if its current status is `VALIDATION_ASKED`. This task is used in *UpdateDocumentsStatus* but can also be used on its own:

```
update_document_status.apply_async((), {"id": id}, eta=eta)
```

create_mangopay_wallet

Takes the id of a `MangoPayWallet` and creates it. See *POST /wallets*.

create_mangopay_pay_out

Takes the id of a `MangoPayPayOut` and creates it. See *POST /payouts/bankwire*.

update_mangopay_pay_out

Takes the id of a `MangoPayPayOut` and updates it. If it still has the status "CREATED" it will be run again the following weekday. See *GET /payouts/{PayOut_Id}*.

CHAPTER 4

Settings

MANGOPAY_CLIENT_ID

The client id you made up when you created your mangopay account.

MANGOPAY_PASSPHRASE

The passphase you recieved when you set up your client.

MANGOPAY_BASE_URL

Set to https://api.mangopay.com in production and https://api.sandbox.mangopay.com for testing.

MANGOPAY_DEBUG_MODE

0 or 1

MANGOPAY_PAGE_DEFAULT_STORAGE

Set this to `True` if you want `MangoPayPage` files to be stored in the default storage. Otherwise you need to have S3BotoStorage set up and working correctly to store the files on AWS.

MANGOPAY_PAYOUT_SUCCEDED_TASK

The name of a task that can be run if and when the payout is successful. It is run immediately with the argument of the *payout_id*.

CHAPTER 5

Contributing

We welcome your contribution.

Please report issues via Githubs Issue tracker.

Please submit pull requests here. All pull requests should include tests. Tests should not touch Mangopay's sandbox only the `MockMangoPayApi` in `tests/client.py`. You must however manually insure all functionality works against Mangopay's sandbox.

To run the tests you will need to install the all requirements including the test specfic ones and then run the test runner.

```
pip install -r requirements.txt
pip install -r requirements_test.txt

./run_tests.py
```

If you make any changes to the documentation you will need to rebuild the docs and commit those changes too.

```
cd docs
make html
```

If you use custom User model in your django project and you would like to run the tests, you will need to create your own User model factory (using factory_boy) and reference it in your django settings in the AUTH_USER_MODEL_FACTORY setting.

Example:

:: AUTH_USER_MODEL_FACTORY = "my_app.factories.UserFactory"

CHAPTER 6

MIT License