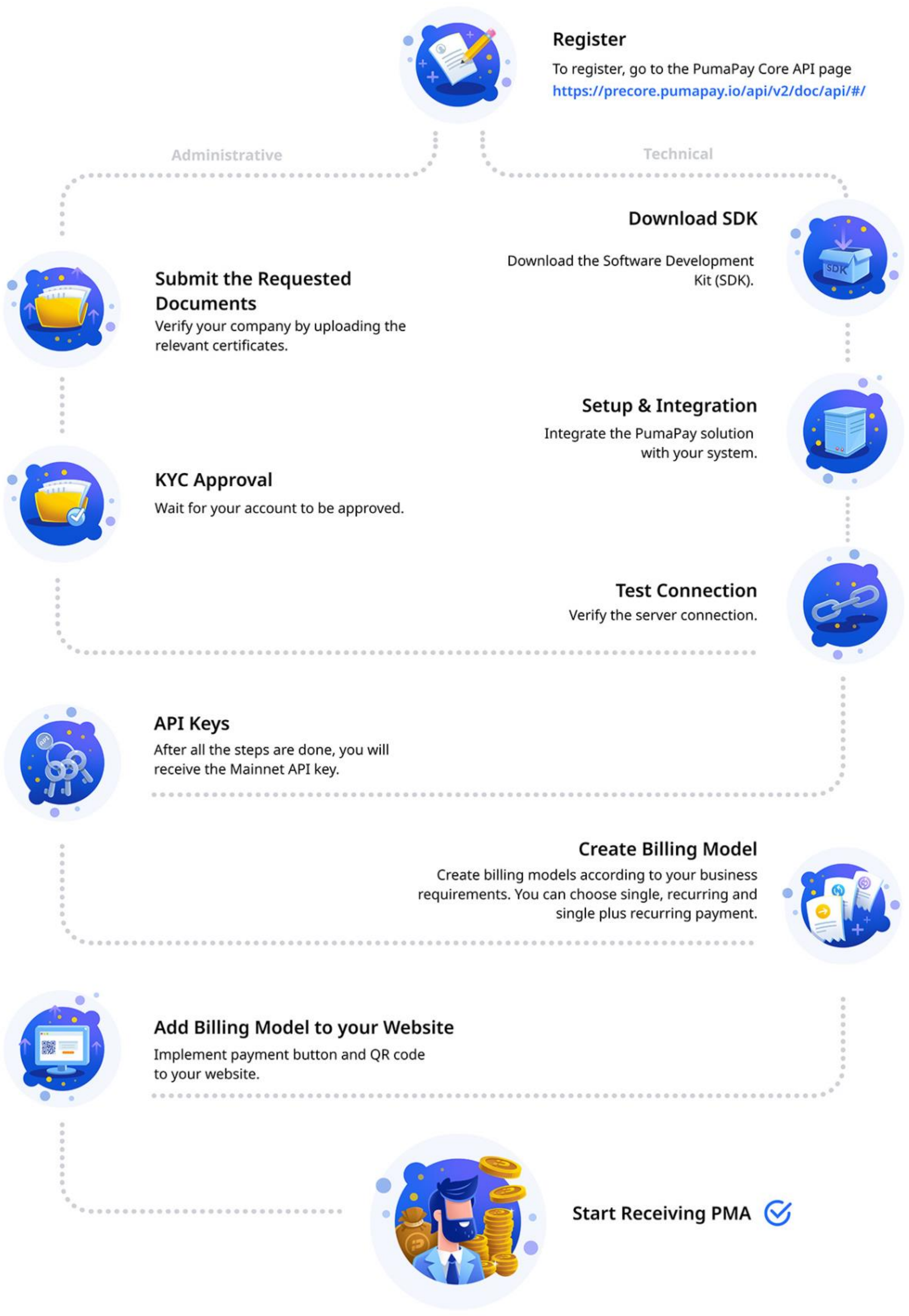
PUMAPAY MERCHANT INTEGRATION GUIDE

DOCUMENTS V.1.2

Merchant Integration Process



Register

To register, go to the PumaPay Core API page https://precore.pumapay.io/api/v2/doc/api/#/

Submit the Requested Documents

Verify your company by uploading the relevant certificates.

KYC Approval

Wait for your account to be approved.

Download SDK

Download the Software Development Kit (SDK).

Setup & Integration

Integrate the PumaPay solution with your system.

Test Connection

Verify the server connection.

API Keys

After all the steps are done, you will receive the Mainnet API key.

Create Billing Model

Create billing models according to your business requirements. You can choose single, recurring and single plus recurring payment.

Add Billing Model to your Website

Implement payment button and QR code to your website.

Start Receiving PMA ✓

MERCHANT INTEGRATION FLOW

The integration process consists of 4 steps
All interactions with PumaPay Core are done via API (Application Programming Interface) calls. For this, software like Postman can be used, as well as the online tools we have in place to perform and test the API calls: https://precore.pumapay.io/api/v2/doc/api/#/

STEP 1 – Registration

1) Register → **2) Log in** → **3) Receive API-Key**

1) Register

At registration you will receive verification email with the verification link

2) Log in

Once logged in, you will receive the PMA X-Token, which will be used to receive your merchants test API-Key

3) Receive API-Key

Once the merchants test API-Key is generated, you are authorized as merchant and can proceed with the integration flow

STEP 2 – Authorization

1) Send Email

In order to become fully authorized merchant, you need to go through KYC process. Please refer to PumaPay KYC procedure. You will send your encrypted data to kyc@pumapay.io

2) Merchant Backend

You will receive the link to the PumaPay Merchant Backend and Merchant Integration documentation

STEP 3 – Setting up the environment

The ideal setup is to build 4 instances working as a single unit. But everything can also be installed and run on a single server. The required runtimes/instances are:

Merchant Backend

The necessary tools to start developing on the PumaPay Pull Payment Protocol and Platform

MySQL

Used for storing the wallets private keys, paying for gas and cashing out PMA

PostgreSQL

Stores the pull payment models, pull payments and Ethereum transactions

Redis

Stores information related to the merchants' address and gas used for the transaction. Also, makes sure if the transaction fails, it can roll back the changes

STEP 4 – Integration and Development

Once everything is set up, you are now ready to use the PumaPay Merchant Backend for development of your own Pull Payment models. For more info, please see the documentation: https://prembackend.pumapay.io/api/v2/doc/api/#/

MERCHANT INTEGRATION GUIDE

PREFACE

PumaPay Pull Payment Protocol is the unique solution allowing the merchants to integrate with PumaPay platform and enabling the recurring payments over the Ethereum blockchain.

Delving deeper into technicalities, the Merchant Portal will give merchants access to a unique set of API keys, which enable communication with the PumaPay server, the blockchain and ultimately, customers' wallets to pull out funds according to predefined terms. A dedicated QR code generator will enable fast and easy sharing of payment details between merchants and shoppers, who authorize the transaction.

ECOSYSTEM OVERVIEW

PumaPay Core
The core is PumaPay custom framework and heart of the system. PumaPay Core allows the governance, control and utilization of the PumaPay Pull Payment Protocol.

PumaPay SDK
The SDK module gives functionality to any third-party integrator to allow execution of pull payments as well as other utility methods.

PumaPay Merchant Backend
The Merchant backend holds a set of APIs to interact with PumaPay ecosystem. It also allows the merchant to have an overview of payment models and payments.

PumaPay Mobile Wallet App
The mobile wallet app allows users who possess PMA to make pull payments with registered merchants who use PumaPay Pull Payment Protocol.

PumaPay Faucet
The faucet is a component that provides PMA test tokens to developers, so that they can test their pull payments after they have integrated with the PumaPay ecosystem.

INTEGRATION FLOW

REQUIREMENTS

To integrate with PumaPay framework you need to have compatible environment and set of tools to proceed with the flow and start the development process.

ENVIRONMENT

a. MULTI-TIER ARCHITECTURE (Recommended Setup)

Ideally you need to setup **4 instances** (servers or cloud servers) which will interact with each other and work as a single unit communicating with PumaPay Core, which is our custom framework and heart of our system, allowing the governance, control and utilization of PumaPay Pull Payment Protocol.

Instance 1: PumaPay Merchant Backend -> the necessary tool to start developing on the PumaPay Pull Payment Protocol and Platform.
Instance 2: MySQL database -> used for storing the wallets private keys, gas payments and cashing out PMA.
Instance 3: PostgreSQL database -> stores the pull payment models, pull payments and Ethereum transactions.
Instance 4: Redis database -> stores information related to the merchants' address and gas used for the transaction. Also, makes the transaction rollback possible if the transaction fails.

Required Software/Runtimes

- **Docker**
- **MySQL**
- **PostgreSQL**
- **Redis**

b. SINGLE TIER ARCHITECTURE WITH ENVIRONMENT SETUP

All of the tools/runtimes will be installed on a single instance/server.

Required Software/Runtimes

Docker, MySQL, PostgreSQL, Redis

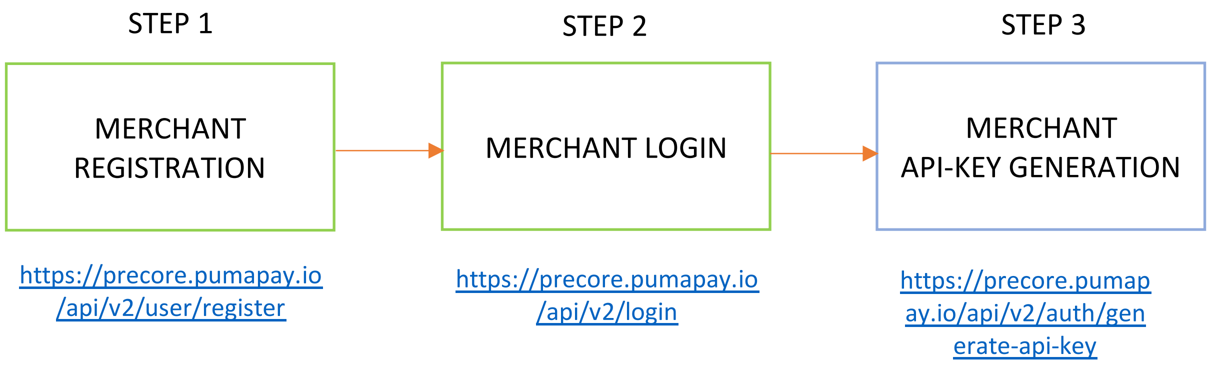
c. SINGLE TIER WITH DOCKER

The simplest and fastest setup with less control over the flow. All of the tools/runtimes will be installed and used from Docker image with additional configurations/data directories.

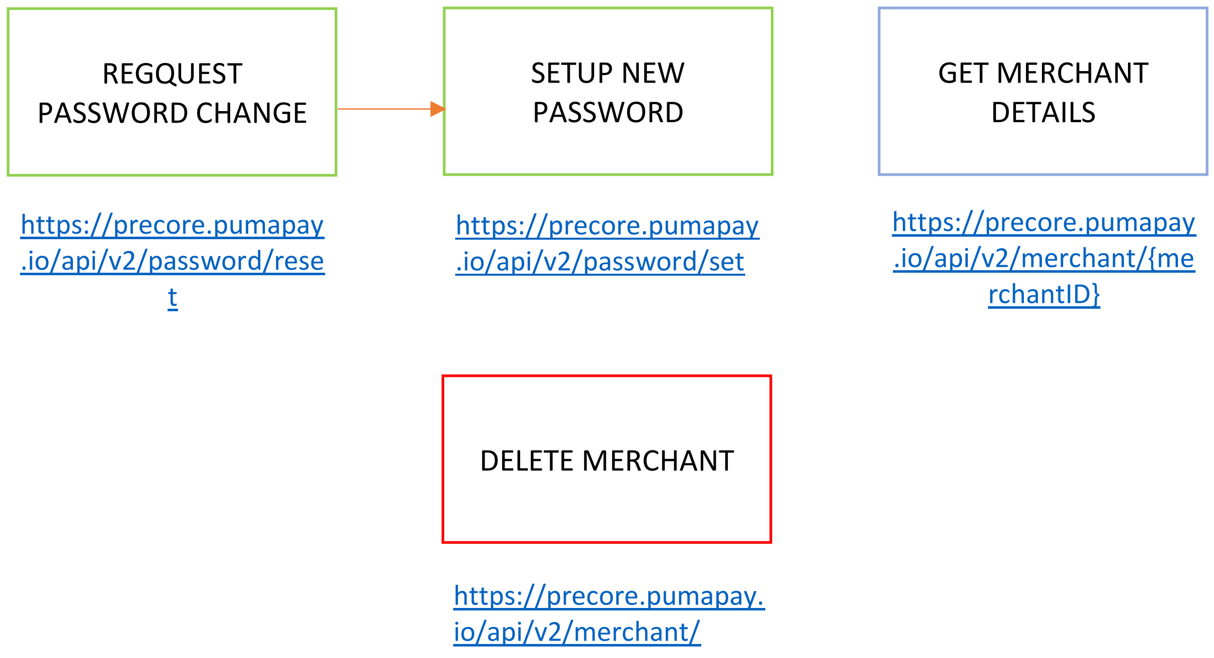
Required Software/Runtimes

Docker

THE FLOW
REGISTRATION PROCESS


STEP 1
MERCHANT REGISTRATION
https://precore.pumapay.io/api/v2/user/register
STEP 2
MERCHANT LOGIN
https://precore.pumapay.io/api/v2/login
STEP 3
MERCHANT API-KEY GENERATION
https://precore.pumapay.io/api/v2/auth/generate-api-key

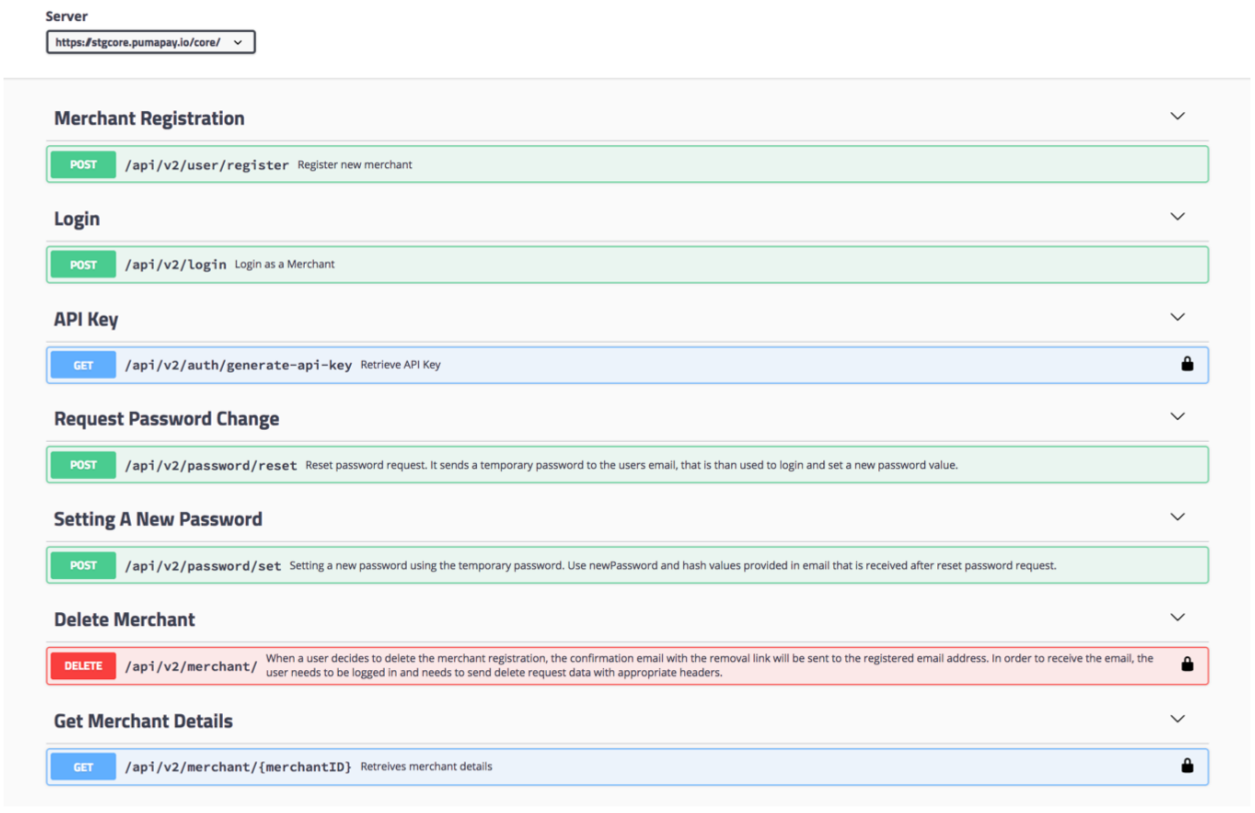
ADDITIONAL FUNCTIONALITY


REGQUEST PASSWORD CHANGE
https://precore.pumapay.io/api/v2/password/reset
SETUP NEW PASSWORD
https://precore.pumapay.io/api/v2/password/set
GET MERCHANT DETAILS
https://precore.pumapay.io/api/v2/merchant/{merchantID}
DELETE MERCHANT
https://precore.pumapay.io/api/v2/merchant/

MERCHANT REGISTRATION

The first thing that the merchant needs to do in order to use the PumaPay Protocol is to register through the PumaPay core APIs in order to retrieve the API key and merchant ID that is essential for setting up their merchant server.

It is possible to use software like Postman for the API calls, but also, we have developed an easy solution for the merchant's registration with the documentation provided.
Navigate here: https://precore.pumapay.io/api/v2/doc/api/#/


Server
https://stgcore.pumapay.io/core/

Merchant Registration
POST /api/v2/user/register Register new merchant

Login
POST /api/v2/login Login as a Merchant

API Key
GET /api/v2/auth/generate-api-key Retrieve API Key

Request Password Change
POST /api/v2/password/reset Reset password request. It sends a temporary password to the users email, that is than used to login and set a new password value.

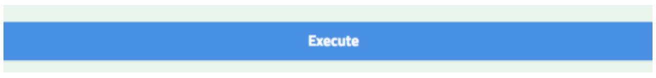
Setting A New Password
POST /api/v2/password/set Setting a new password using the temporary password. Use newPassword and hash values provided in email that is received after reset password request.

Delete Merchant
DELETE /api/v2/merchant/ When a user decides to delete the merchant registration, the confirmation email with the removal link will be sent to the registered email address. In order to receive the email, the user needs to be logged in and needs to send delete request data with appropriate headers.

Get Merchant Details
GET /api/v2/merchant/{merchantID} Retreives merchant details

INTERFACE EXPLANATION

To access each function, use the button:

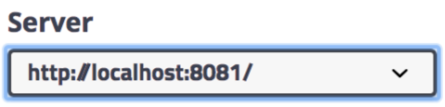
To execute functions, use the button: 

Some functions require authorization, which is identified by: 🔒

It is also possible to use software like Postman or Curl to call the Merchant Core APIs.
API calls examples, as well as Curl examples are provided in the swagger documentation.

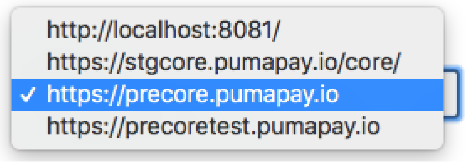
Current document describes how to do it with Swagger, provided with the Merchant Core APIs.

From the server selection menu make sure
to select the **PRECORE**

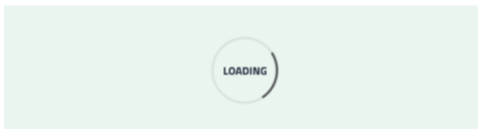


Server

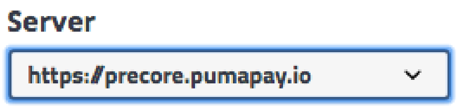
http://localhost:8081/



http://localhost:8081/
https://stgcore.pumapay.io/core/
✓ https://precore.pumapay.io
https://precoretest.pumapay.io



This icon will always run on execute and will disappear once the process has completed

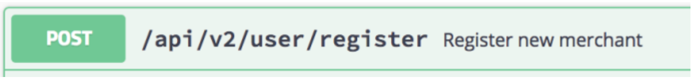


Server

https://precore.pumapay.io

Click on the names/links in colored boxes to expand and access the available functions.

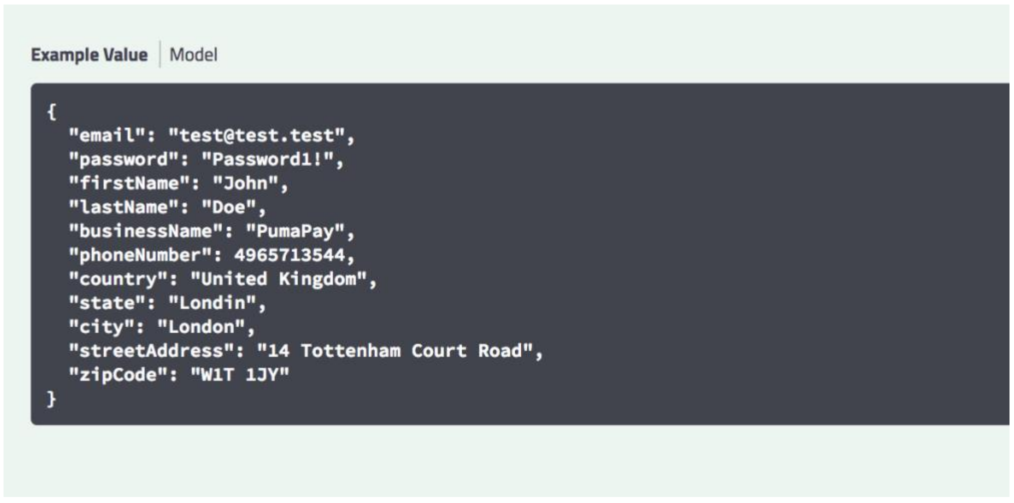
POST -> MERCHANT REGISTRATION


POST /api/v2/user/register Register new merchant

https://precore.pumapay.io/api/v2/user/register
(please note: if using this function with third-party software to access the PumaPay Core, use the above address with POST method and Body described below)

To get access to our Cores' functionality and to integrate, firstly, you need to register with our Core API as a merchant using the POST request and application/json data in the body of the call.

Use request body like in the example below, to register with your merchant credentials:



```
{
  "email": "test@test.test",
  "password": "Password1!",
  "firstName": "John",
  "lastName": "Doe",
  "businessName": "PumaPay",
  "phoneNumber": 4965713544,
  "country": "United Kingdom",
  "state": "Londin",
  "city": "London",
  "streetAddress": "14 Tottenham Court Road",
  "zipCode": "W1T 1JY"
}
```

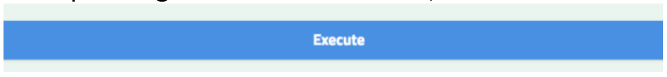
Click on the "Try It Out" button

Inside the BODY of the API call (model window) you need to provide the information as shown above, where:

Email: is the email that you will use for registration and verification with PumaPay

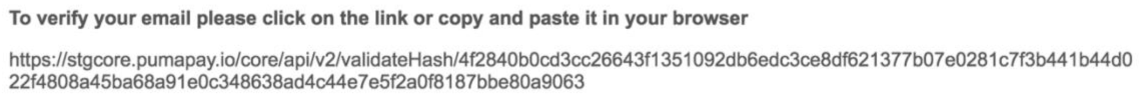
Password: a unique secure password that will allow you to login into our system
*Please note that **password must include** the **combination** of at least one **lower case** letter, one **upper case**, one **number** and one **special character** (ex.: !@#$ etc.)*

firstname: is your first name
lastname: is your last name
businessname: is the name of your business/company
phonenumber: is your contact phone number
country, state, city, streetaddress, zipcode: form the address of your business

After pressing the "execute" button,


you will see the "loading" message and on successful execution you should get response (code 200) shown below the form you have.
In case if the request fails and you don't receive the confirmation email, please contact PumaPay support.

Once succeeded, you should get successful response and receive an email with verification link, as in the example below.



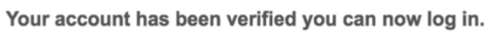
To verify your email please click on the link or copy and paste it in your browser

https://stgcore.pumapay.io/core/api/v2/validateHash/4f2840b0cd3cc26643f1351092db6edc3ce8df621377b07e0281c7f3b441b44d0
22f4808a45ba68a91e0c348638ad4c44e7e5f2a0f8187bbe80a9063

Following/copy the link and paste it into the browser of your choice. It will generate a response, similar to the one below:

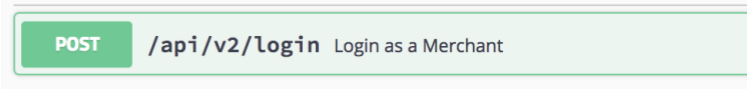
```
{
    "success":true,
    "status":200,
    "message":"User Verified.",
    "data":
        [{
        "verificationHashID":"4ab56670-cdf8-11e8-b361-d386a01ecd93",
         "hash":"4f2840b0cd4cc266TEST351092db6edc3ce8df621377b07e0281c7f3b441b44d
022f4808a45ba68a91e012348ad4c44e7e5f2a0f8187bbe80a9063",
        "userID":"4ab1cb28-cdf8-11e8-8564-6ff6c36a8e82",
        "timeCreated":"1539332692"
    }]
}
```

As well as you will receive another email confirming that the account has been successfully verified:



Your account has been verified you can now log in.

POST -> MERCHANT LOGIN

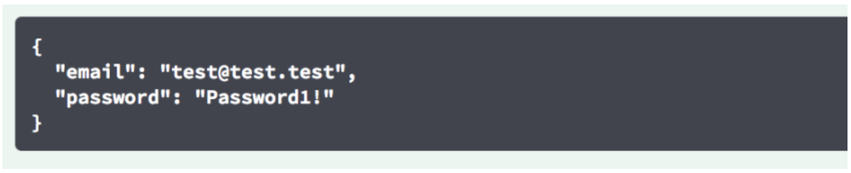

POST /api/v2/login Login as a Merchant

https://precore.pumapay.io/api/v2/login

By using the credentials, you have registered with, you can login after your registration and account have been verified.

Once you'll login, you will be assigned with the new generated merchant authorization token that will allow you to receive the **merchant test API-Key** to access the PumaPay Core API functionality and test on the Merchant Backend. To receive the merchant's live API-Key, you need to pass our **KYC procedure** (for more information please refer to the PumaPay KYC procedure).

Use your registration credential to call the API with the body as in the example below:

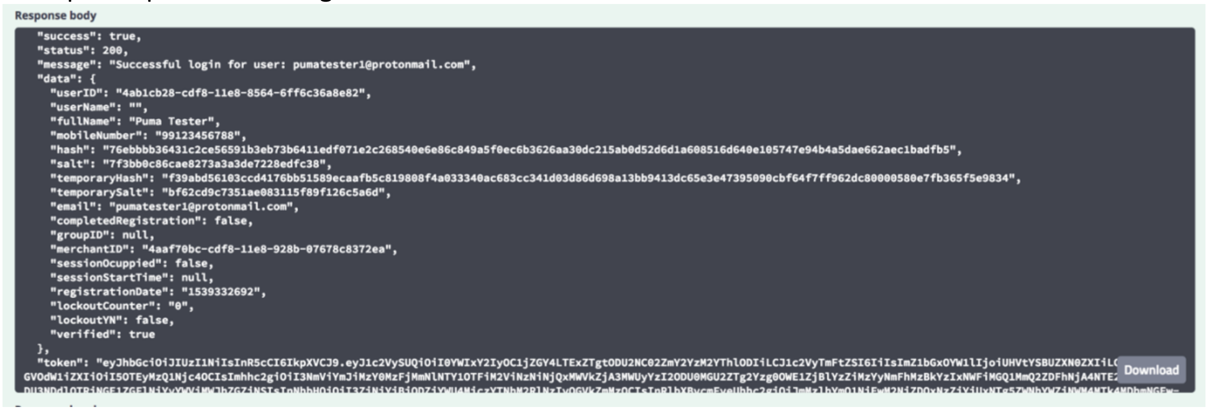

```
{
  "email": "test@test.test",
  "password": "Password1!"
}
```

Successful response:

Make sure to save **merchantID** and **Token,** you will need them later.

Token will be used to generate your Merchant test API-Key, which is the next step, to access the core functionality of PumaPay system.

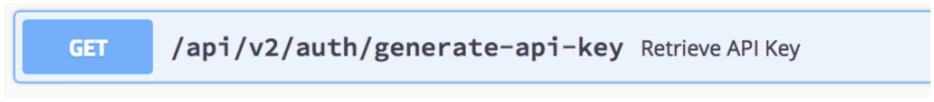
Example response including the Token:


Response body

```
  "success": true,
  "status": 200,
  "message": "Successful login for user: pumatester1@protonmail.com",
  "data": {
    "userID": "4ab1cb28-cdf8-11e8-8564-6ff6c36a8e82",
    "userName": "",
    "fullName": "Puma Tester",
    "mobileNumber": "99123456788",
    "hash": "76ebbbb36431c2ce56591b3eb73b6411edf071e2c268540e6e86c849a5f0ec6b3626aa30dc215ab0d52d6d1a608516d640e105747e94b4a5dae662aec1badfb5",
    "salt": "7f3bb0c86cae8273a3a3de7228edfc38",
    "temporaryHash": "f39abd56103ccd4176bb51589ecaafb5c819808f4a033340ac683cc341d03d86d698a13bb9413dc65e3e47395090cbf64f7ff962dc80000580e7fb365f5e9834",
    "temporarySalt": "bf62cd9c7351ae083115f89f126c5a6d",
    "email": "pumatester1@protonmail.com",
    "completedRegistration": false,
    "groupID": null,
    "merchantID": "4aaf70bc-cdf8-11e8-928b-07678c8372ea",
    "sessionOcuppied": false,
    "sessionStartTime": null,
    "registrationDate": "1539332692",
    "lockoutCounter": "0",
    "lockoutYN": false,
    "verified": true
  },
  "token": "eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJ1c2VySUQiOiI0YWIxY2IyOC1jZGY4LTExZTgtODU2NC02ZmY2YzM2YThlODIiLCJ1c2VyTmFtZSI6IiIsImZ1bGxOYW1lIjoiUHVtYSBUZXN0ZXIiLC
GVOdW1iZXIiOiI5OTEyMzQ1Njc4OCIsImhhc2giOiI3NmViYmJiMzY0MzFjMmNlNTY1OTFiM2ViNzNiNjQxMWVkZjA3MWUyYzI2ODU0MGU2ZTg2Yzg0OWE1ZjBlYzZiMzYyNmFhMzBkYzIxNWFiMGQ1MmQ2ZDFhNjA4NTE
[illegible]
```
Download

Copy/Save generated merchants **token,** so you can use it in the next step to generate Merchant test API-Key.

GENERATE MERCHANT API-KEY

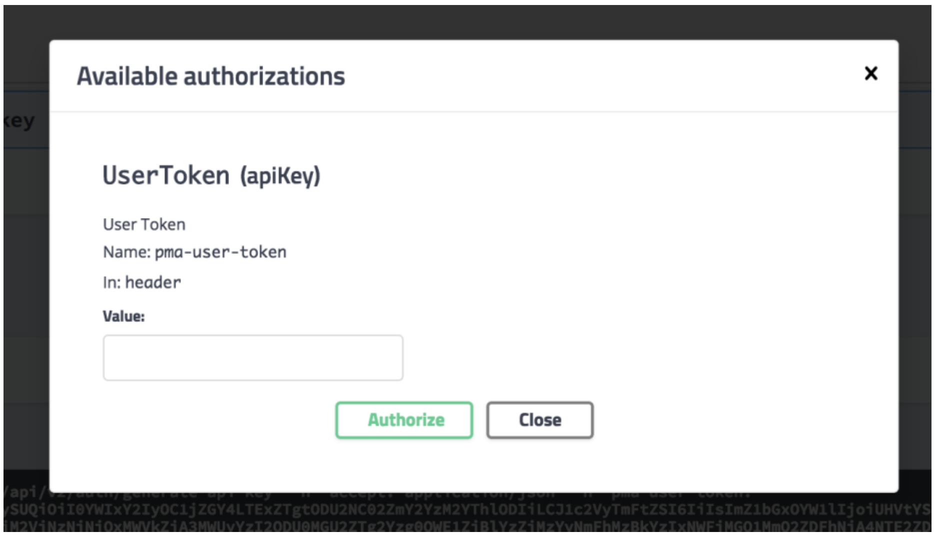

GET /api/v2/auth/generate-api-key Retrieve API Key

https://precore.pumapay.io/api/v2/auth/generate-api-key

Use the unlock button to access the functionality:

*(if you are using the third-party software, use `pma-user-token` variable in the call HEADERS with the **token** generated at login and using the GET method)*

Paste the **Token** generated in the previous step into the **value** field, click **Authorize**, once you see the **value** turning into ******, you can click **Close**:

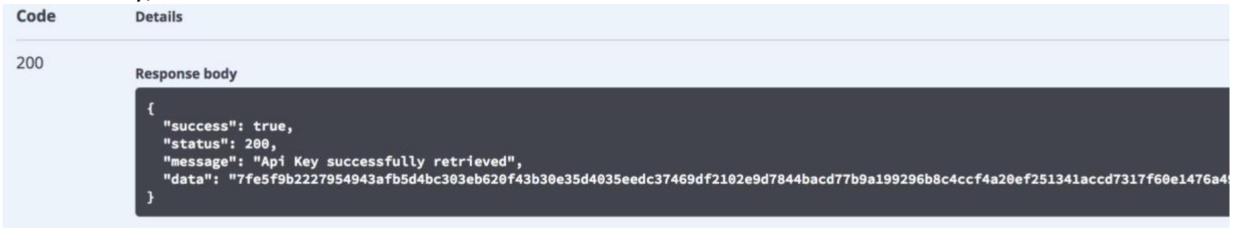

Available authorizations

UserToken (apiKey)

User Token
Name: pma-user-token
In: header
Value:

After that, if you have been successfully authorized, press the “Try it out” button, and then “Execute”.

On successful execution, in a few seconds you will receive the response containing your Merchants’ test API-Key, similar to this:

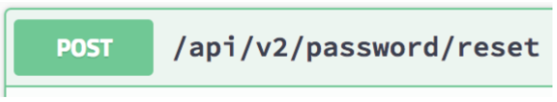


```
{
  "success": true,
  "status": 200,
  "message": "Api Key successfully retrieved",
  "data": "7fe5f9b2227954943afb5d4bc303eb620f43b30e35d4035eedc37469df2102e9d7844bacd77b9a199296b8c4ccf4a20ef251341accd7317f60e1476a4
}
```

The data field is the actual API-Key you will use to get authorized by the PumaPay Core and it will allow you to interact with the whole ecosystem and perform all the operations.

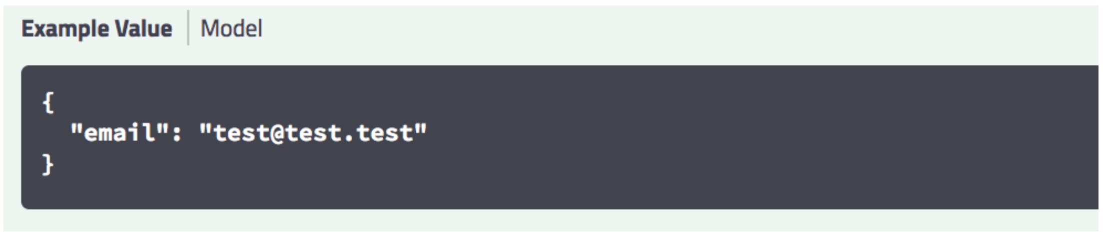
At this stage you are ready to use our Merchant Backend.

POST -> REQUEST PASSWORD CHANGE



https://precore.pumapay.io/api/v2/password/reset

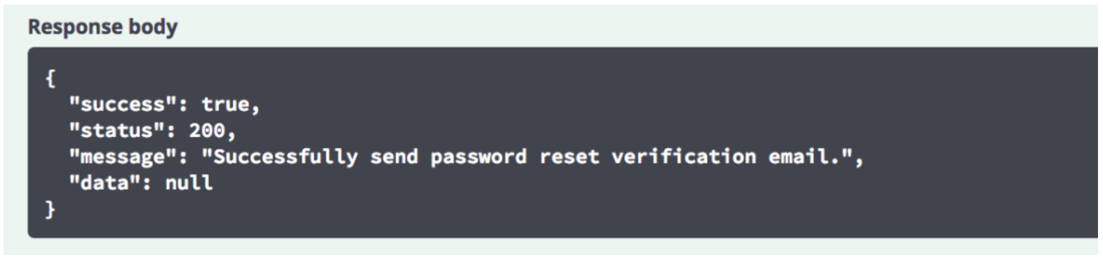
If for any reason you need to change the password, execute/call this function using the email you registered with in the BODY (MODEL) as in the example below:



```
{
  "email": "test@test.test"
}
```

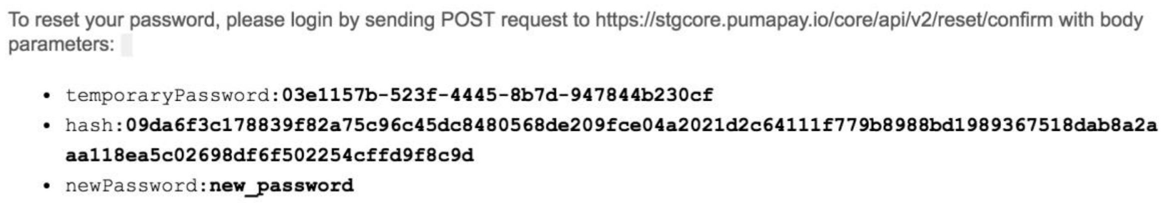
Use the “Try it out” button, replace the data inside the model with your registration email and press execute.

This will generate and send the password reset link to your email as well as you should see the following response



```
{
  "success": true,
  "status": 200,
  "message": "Successfully send password reset verification email.",
  "data": null
}
```

In the received email you should see the data similar to the one below:



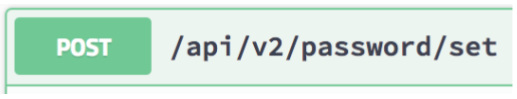
To reset your password, please login by sending POST request to https://stgcore.pumapay.io/core/api/v2/reset/confirm with body parameters:

- temporaryPassword:**03e1157b-523f-4445-8b7d-947844b230cf**
- hash:**09da6f3c178839f82a75c96c45dc8480568de209fce04a2021d2c64111f779b8988bd1989367518dab8a2aaa118ea5c02698df6f502254cffd9f8c9d**
- newPassword:**new_password**

This means that the new temporary password was generated, and you are authorized to change the password.

Copy the **temporaryPassword** and **hash** in order to change the password and proceed to the **SETTING A NEW PASSWORD** section/function

POST -> SETTING A NEW PASSWORD



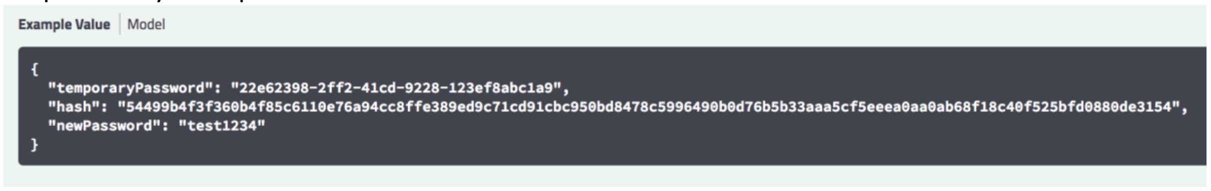
```
/api/v2/password/set
```

https://precore.pumapay.io/api/v2/password/set

Setting a new password using the temporary password.

Use **temporaryPassword**, **hash** from the email you have received after executing the *REQUEST PASSWORD CHANGE* and in the BODY of the API call use those values, plus the newPassword, in order to change the password.

Request body example:



```
{
  "temporaryPassword": "22e62398-2ff2-41cd-9228-123ef8abc1a9",
  "hash": "54499b4f3f360b4f85c6110e76a94cc8ffe389ed9c71cd91cbc950bd8478c5996490b0d76b5b33aaa5cf5eeea0aa0ab68f18c40f525bfd0880de3154",
  "newPassword": "test1234"
}
```

Make sure to use all values as strings (wrap them in comments “ “).
On successful request, you will receive code 200 and verification message that the temporary password is correct.

Now your password has been changed to the **newPassword** and you can use it to login from now on.

DELETE -> DELETE MERCHANT

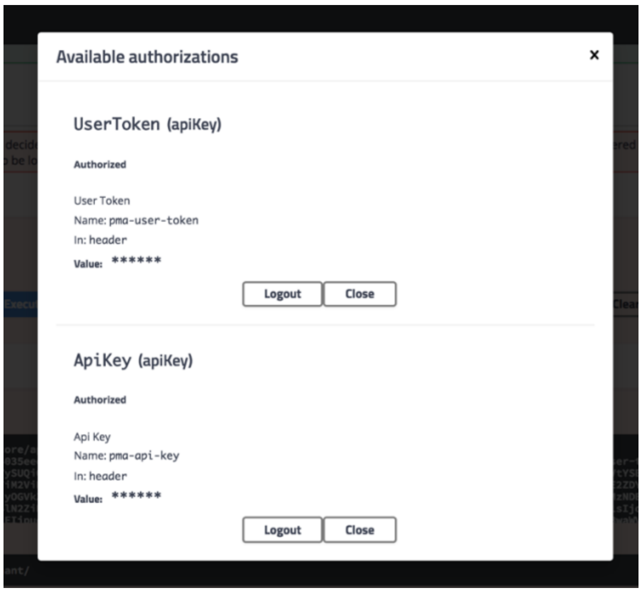
DELETE /api/v2/merchant/

https://precore.pumapay.io/api/v2/merchant/
(please note: DO NOT experiment with this unless you really wish to delete your account)

When a user decides to delete the merchant registration, the confirmation email with the removal link will be sent to the registered email address.

In order to receive the email, the user needs to be logged in 🔒 using the token you have received on login and the merchant API-Key.

Please note: when using the thirid-party sogtware, use the `pma-user-token` and the `pma-api-key` in the HEADERS of the API call


Available authorizations ×

UserToken (apiKey)

Authorized

User Token
Name: pma-user-token
In: header
Value: ******

ApiKey (apiKey)

Authorized

Api Key
Name: pma-api-key
In: header
Value: ******

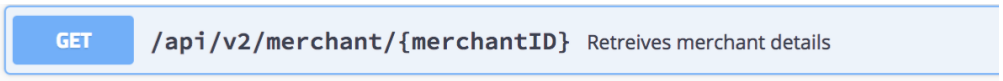
Once executed, you will receive the email with the link to confirm the deletion of your account.

Follow/copy and paste this link to delete your merchant account.

Please note: once the deletion link from the email is followed, no further confirmations will be asked, and your account will be removed from our system.

WARNING: THIS WILL DELETE THE MERCHANTS ACCOUNT AND ALL OF THE MERCHANTS DATA

GET -> GET MERCHANT DETAILS



/api/v2/merchant/{merchantID} Retreives merchant details

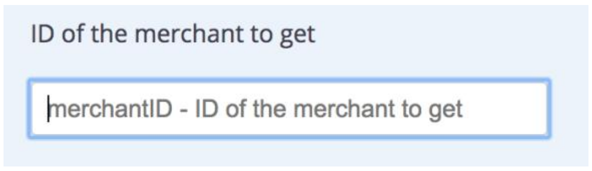
https://precore.pumapay.io/api/v2/merchant/{merchantID}

Retrieves merchant details.

You need to be logged in and your user token should be authorized (if not, login or use the generated token from the login) as well as the merchants API-Key.

(Please note: when using third-party software, you will use the **merchantID** inside the API call link and inside the HEADERS you will use `pma-user-token` and the `pma-api-key`)

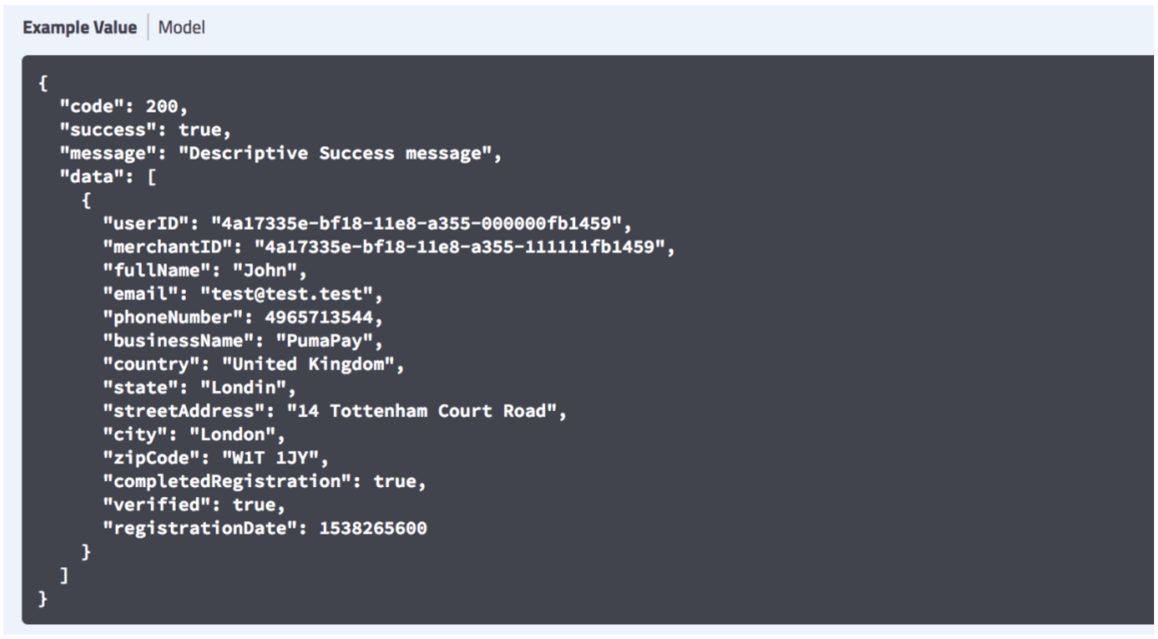
Once logged in (you can check the status by pressing on the lock icon) use “Try it out button” and provide your merchant id,



ID of the merchant to get

merchantID - ID of the merchant to get

then press “execute”. After execution, the system should respond with a similar response as shown below with your merchant details:



```
{
  "code": 200,
  "success": true,
  "message": "Descriptive Success message",
  "data": [
    {
      "userID": "4a17335e-bf18-11e8-a355-000000fb1459",
      "merchantID": "4a17335e-bf18-11e8-a355-111111fb1459",
      "fullName": "John",
      "email": "test@test.test",
      "phoneNumber": 4965713544,
      "businessName": "PumaPay",
      "country": "United Kingdom",
      "state": "Londin",
      "streetAddress": "14 Tottenham Court Road",
      "city": "London",
      "zipCode": "W1T 1JY",
      "completedRegistration": true,
      "verified": true,
      "registrationDate": 1538265600
    }
  ]
}
```

MERCHANT AUTHORIZATION

After you have successfully registered as a merchant, you need to complete the KYC authorization procedure

PUMAPAY KYC PROCEDURE

To become fully authorized merchant, you need to go through the KYC procedure.
Please make sure to send your documents, encrypted with the software/tool of your choice.
We recommend PGP Desktop, we will send you our public key in order to encrypt the necessary documentation.
Please send encrypted document specified in the list below to kyc@pumapay.io

Merchant is requested to submit the below list of documents for verification.

1. Certificate of incorporation
2. Certificate of a) shareholders, b) directors and c) registered office or Incumbency certificate that includes all this information
3. Utility bills of shareholders and directors (which are not more than 3 months old)
4. Passport of shareholder and directors
5. Memorandum and Articles of Association

Our compliance officer will review submitted documentation and may contact you if we require additional documentation.

Verification process

1. We are running the documents through our internal Verification process
2. Once approved, we are changing the Merchant's status to Verified
3. Merchant can continue and receive the API keys.

Once we receive the email, we will respond within 24 hours on working days with the link to the Docker image.

The PumaPay Docker image is required in every case, but the setup will be different and based on your preference (as described in the **Requirements** section) you will be able to use the dockerfile from *Merchant Integration Wiki* to setup the servers.

Once we will receive your Docker Hub username/email we will share our private repository.
If you don't have Docker Hub account, you can register at: https://hub.docker.com/

CHOISE OF SERVERS

We recommend using dedicated servers/instances to install and prepare all required dependencies/tools/runtimes to integrate with PumaPay Pull Payment Protocol.

We use AWS – Amazon Web Services.

Also, our Merchant Backend supports AWS Encryption

REQUIRED TOOLS

Please note that it is not recommended to use the root user to install the software and setup the servers. It is recommended to you use another user account, not root.

The following tools/runtimes are required.
Note: if you need detailed guide on how to install the tools, follow our HOWTO documentation.

1. **DOCKER**
2. **MYSQL**
3. **POSTGRESQL**
4. **REDIS**
5. **PUMAPAY MERCHANT BACKEND**

Merchant Backend can be installed in 3 possible ways:

- on a multi-tier architecture (4 instances),
- on a single instance,
- all on a single instance, running from docker images

SETTING UP THE SERVERS

From the Merchant Guide Wiki you will be able to find docker setup file to configure the instance of your Merchant Backend in order to function and correctly communicate with the PumaPay Core.

On the instance containing the Docker image with the PumaPay Merchant Backend, create the **`docker-compose.yml`** file.

```
touch docker-compose.yml
```

We *don't recommend* using the *root* user setting up the environment, it's best to use a user that will have access to sudo, but won't have the super admin permissions.

configure the docker file with the following parameters:
Please note: the configuration of your docker file will depend on the type of the tier setup you will choose, but the main credentials will remain the same for all 3 types of PumaPay Merchant Backend setup.

```
- NODE_ENV=development                         # development for testnet /
production for testnet
- HOST=localhost                               # server host
- PORT=3000                                    # server port
- CORE_API_URL=https://precore.pumapay.io/core  # PumaPay core URL
- MERCHANT_URL=http://localhost:3000           # Merchant server URL
- PGHOST=postgres_host                     # PostgreSQL db host
- PGUSER=db_user                               # PostgreSQL db user
- PGPASSWORD=db_pass                           # PostgreSQL db password
- PGDATABASE=db_name                           # PostgreSQL db name
- PGPORT=5432                                  # PostgreSQL db port
- REDIS_PORT=6379                              # Redis Port
- REDIS_HOST=redis_host                    # Redis Host
- REDIS_TOKEN=123456789                        # Redis token - AWS Setup
- ETH_NETWORK=3                                # Ethereum network - 3 for testnet
/ 1 for mainnet
- KEY_DB_HOST=db_host                          # MySQL db host
- KEY_DB_USER=db_user                          # MySQL db user
- KEY_DB_PASS=db_pass                          # MySQL db password
- KEY_DB=db_name                               # MySQL db name
- KEY_DB_PORT=3306                             # MySQL db port
- MNEMONIC_ID=mnemonic_phrase_id               # Mnemonic phrase ID - as stored
in MySQL db from the SQL script
- BALANCE_MONITOR_INTERVAL=21600000            # Time interval in seconds to
monitor the balance of the bank wallet account
- BALANCE_CHECK_THRESHOLD=0.1                  # Threshold which will send an
email notifcation to the email provided
- SENDGRID_API_KEY=RETRIEVE_ONE_FROM_SENDGRID   # SendGrid API key - is used for
sending emails related with wallet balances
- BALANCE_CHECK_EMAIL=test@test.test           # Receiver email for the balance
checker - testing environment
- BALANCE_CHECK_EMAIL_PROD=test@test.test      # Receiver email for the balance
checker - production environment
- CORE_API_KEY=API_KEY_RETRIEVED_FROM_CORE      # API key retrieved after
registering to PumaPay core server
- MERCHANT_ID=MERCHANT_ID_RETRIEVED_FROM_CORE   # Merchant ID as retrieved from
PumaPay core after registration
# For AWS, you can change the ENCRYPTION_MODULE from none to AWS and specify AWS
credentials
- AWS_ACCESS_KEY_ID=0
- AWS_SECRET_ACCESS_KEY=0
- AWS_REGION=0
- AWS_KEY_ID=0
- ENCRYPTION_MODULE=none
```

You will need to enter the **credentials** you have created for each **database** user during the tools/runtimes **setup** and the hostname will be based on the tier architecture. For the **multi-tier architecture,** you will need to specify the **IP address** of each instance, where in **single-tier architecture** and docker instance it will be the **localhost**

Once the docker file is configured, you can run the following command to start the instance of the PumaPay Merchant Backend instance:

```
docker-compose up -d
```

if everything is setup correctly, the instance will run, and you will be able to communicate with it using PumaPay Merchant Backend API calls.

Please refer to this documentation for more info:
https://prembackend.pumapay.io/api/v2/doc/api/#/
For more technical information and setup guide please refer to the PumaPay Merchant Wiki in our GitHub repository: https://github.com/pumapayio/wiki