pyjengahq

Tralah M Brian

Jun 22, 2021

CONTENTS:

PyJengaHQ is a python library to help python programmers interact with Equity's Jenga APIs.

CHAPTER

ONE

GETTING STARTED

1.1 Installation

The package is available on pypi.org <https://pypi.org/projects/pyjengahq>.

```
pip install pyjengahq
```

1.2 First Things First

1.2.1 Generating an openSSL private/public key pair.

Since **v2** of the API, JengaHQ now requires some APIs to include signatures as a security measure. This signatures are usually generated by signing some request fields with a private key. In order to use JengaHQ you'll have to upload your public key to the developer console on JengaHQ after you generate your key pair.

This Library provides a CLI command to help you generate those keys.

Note: The generated keys will be stored in your home directory under the *.JengaAPI/keys/* folder and this is where by default the Library looks for your private key when initiating JengaRequest Objects.

After Installation, Generate Your Keys using the following command:

```
$ jenga_gen_key_pair
```

Note: Then copy the *~/.JengaApi/keys/publickey.pem* to your developer console on JengaHq.

1.3 Using The APIs

1.3.1 Creating JengaAPI objects

By Design, interaction with all Jenga APIs functionality uses the *jengahq.auth.JengaAPI* class which takes the following parameters:

- api_key: Your Jenga API Key
- password: Your Jenga API Password

- merchant_code:: the merchant code provided by JengaHQ
- env:: the environment in which the API is to be used either *sandbox* or *production*
- private_key:: the path to the merchant private key default is "~/.JengaAPI/keys/privatekey.pem"
- sandbox_url:: the url used to access the Sandbox API
- live_url:: the url used to access the Production API

Example

```
import jengahq
jengaApi = jengahq.JengaAPI(
    api_key="Basic TofFGUeU9y448idLCKVAe35LmAtLU9y448idLCKVAe35LmAtL",
    password="TofFGUeU9y448idLCKVAe35LmAtL",
    merchant_code="4144142283",
    env="sandbox",
)
```

CHAPTER

TWO

JENGAHQ API REFERENCE

2.1 jengaAPI

Authentication module.

class jengahq.auth.**JengaAPI**(*api_key: str*, *password: str*, *merchant_code: str*, *env: str = 'sandbox'*, *private_key='/home/docs/.JengaApi/keys/privatekey.pem'*, *sandbox_url: str = 'https://sandbox.jengahq.io'*, *live_url: str = 'https://api.jengahq.io'*)

Bases: `object`

Jenga API CORE Class Representation.

Jenga Payment Gateway and Jenga API support the OAuth 2.0 Authentication Framework, requiring you to provide a username and password, as well as an API key that you generate on Jenga HQ part of HTTP Basic Authentication to generate a Bearer token.

Once you have a token you can make subsequent requests to initiate payments, check completed transactions and more.

Only load your API keys as environment variables and do not share your credentials to anyone over email or any other method of communication.

Params

Api_key Your Jenga API Key

Password Your Jenga API Password

Merchant_code: the merchant code provided by JengaHQ

Env: the environment in which the API is to be used either *sandbox* or *production*

Private_key: the path to the merchant private key default is "~/.JengaAPI/keys/privatekey.pem"

Sandbox_url: the url used to access the Sandbox API

Live_url: the url used to access the Production API

Example

```
import jengahq
jengaApi = jengahq.JengaAPI(
    api_key="Basic TofFGUeU9y448idLCKVAe35LmAtLU9y448idLCKVAe35LmAtL",
    password="TofFGUeU9y448idLCKVAe35LmAtL",
    merchant_code="4144142283",
    env="sandbox",
)
```

property authorization_token: str
Return Authorization Token.

Returns a str like to be used in header as Authorization value

```
"Bearer ceTo5RCpluTfGn9B3OZXnnQkDVKM"
```

get_account_available_balance(*countryCode: str*, *accountId: str*) → dict
Get Account's available balance.

Retrieve the current and available balance of an account

200 Success Response Schema

```
{
    "currency": "KES",
    "balances": [
        {
            "amount": "997382.57",
            "type": "Current"
        },
        {
            "amount": "997382.57",
            "type": "Available"
        }
    ]
}
```

get_account_full_statement(*countryCode: str*, *accountNumber: str*, *fromDate: str*, *toDate: str*, *limit=10*)
Get account full statement.

Example Response

```
{
    "accountNumber": "0011547896523",
    "currency": "KES",
    "balance": 997382.57,
    "transactions": [
        {
            "reference": 541,
            "date": "2018-07-13T00:00:00.000",
            "description": "EQUITEL-BUNDLE/254764555383/8755",
            "amount": 900,
            "serial": 1,
            "postedDateTime": "2018-07-13T09:51:27.000",
            "type": "Debit",
            "runningBalance": {
                "currency": "KES",
                "amount": 1344.57
            }
        },
        {
            "reference": "S4921027",
            "date": "2018-07-18T00:00:00.000",
            "description": "EAZZY-AIRTIME/EQUITEL/254764555383/100000939918/18",
```

(continues on next page)

(continued from previous page)


```
            "amount": 200,
            "serial": 1,
            "postedDateTime": "2018-07-18T16:27:18.000",
            "type": "Debit",
            "runningBalance": {
                "currency": "KES",
                "amount": 1144.57
            }
        },
        {
            "reference": 5436,
            "date": "2018-07-19T00:00:00.000",
            "description": "CREDIT TRANSFER",
            "amount": 1000000,
            "serial": 2,
            "postedDateTime": "2018-07-19T12:01:47.000",
            "type": "Credit",
            "runningBalance": {
                "currency": "KES",
                "amount": 1001144.57
            }
        }
    ]
}
```


get_account_mini_statement(*countryCode: str*, *accountNumber: str*)

Get account mini statement.

Example Response


```
{
    "accountNumber": "0011547896523",
    "currency": "KES",
    "balance": 1000,
    "transactions": [
        {
            "chequeNumber": null,
            "date": "2017-01-01T00:00:00",
            "description": "EAZZY-FUNDS TRNSF TO A/C XXXXXXXXXXXX",
            "amount": "100",
            "type": "Debit"
        },
        {
            "chequeNumber": null,
            "date": "2017-01-03T00:00:00",
            "description": "SI ACCOUNT TO ACCOUNT THIRD PA",
            "amount": "51",
            "type": "Debit"
        },
        {
            "chequeNumber": null,
            "date": "2017-01-05T00:00:00",
            "description": "CHARGE FOR OTC ECS TRAN",
```


(continues on next page)

(continued from previous page)


```
            "amount": "220",
            "type": "Debit"
        },
        {
            "chequeNumber": null,
            "date": "2017-01-05T00:00:00",
            "description": "SI ACCOUNT TO ACCOUNT THIRD PA",
            "amount": "20",
            "type": "Debit"
        }
    ]
}
```


get_account_opening_and_closing_balance(*accountId: str*, *countryCode: str*, *date: str*)
Get account opening and closing balance.

Example Request


```
{
"countryCode": "KE",
"accountId": "0011547896523",
"date": "2017-09-29"
}
```


Example Response


```
{
    "balances": [
        {
            "type": "Closing Balance",
            "amount": "10810.00"
        },
        {
            "type": "Opening Balance",
            "amount": "103.00"
        }
    ]
}
```


get_all_billers(*numPages: int = 1*, *per_page: int = 10*)
Return all billers.

This web service returns a paginated list of all billers

get_all_eazzypay_merchants(*numPages: int = 1*, *per_page: int = 10*)
Return all EazzyPay merchants.

This webservice returns all EazzyPay merchants .

get_forex_rates(*countryCode: str*, *currencyCode: str*) → dict
Return Forex rates.

Params

CountryCode: the country for which rates are being requested. Valid values are KE, TZ, UG, RW.

CurrencyCode: the currency code of the currency that is being converted from in ISO 4217 format

Example Request

```
{
"countryCode": "KE",
"currencyCode": "USD"
}
```

Example Response :currencyRates:: list of conversion rates for major currencies

```
{
"currencyRates":[],
"fromCurrency": "KES",
"rate":101.3,
"toCurrency": "USD"
}
```

get_payment_status(*transactionReference: str*)

Return payment status.

The webservice enables an application track the status of a payment that is linked to the Receive Payments - Eazzypay Push web service especially in failure states.

get_pesalink_linked_accounts(*mobile_number: str*)

Return pesalink lined accounts.

This webservice returns the recipients' Linked Banks linked to the provided phone number on PesaLink

get_transaction_details(*transactionReference: str*)

Return transaction details.

This webservice enables an application or service to query a transactions details and status

get_transaction_status(*requestId: str*, *transferDate: str*)

Get transaction status.

Use this API to check the status of a B2C transaction

kyc_search_verify(*identity: dict*)

Know your customer search and verify.

Params:

Identity

documentType string the document type of the customer. for example ID, PASSPORT, ALIENID

firstName string first name as per identity document type

lastName string last name as per identity document type

dateOfBirth string optional date in YYYY-MM-DD format

documentNumber string the document id number

countryCode string the country in which the document relates to (only KE and RW enabled for now)

Example Reponse

```
{
    "identity": {
        "customer": {
            "fullName": "John Doe ",
            "firstName": "John",
            "middlename": "",
            "lastName": "Doe",
            "ShortName": "John",
            "birthDate": "1900-01-01T00:00:00",
            "birthCityName": "",
            "deathDate": "",
            "gender": "",
            "faceImage": "/9j/4AAQSkZJRgABAAEAYABgA+H8qr6n4e1O71SGFbV/sEOF3O6/N/
↪eb71d/FGkaBVXaq9KfRRRRRUMsKSIdyr0r/9k=",
            "occupation": "",
            "nationality": "Refugee"
        },
        "documentType": "ALIEN ID",
        "documentNumber": "654321",
        "documentSerialNumber": "100500425",
        "documentIssueDate": "2002-11-29T12:00:00",
        "documentExpirationDate": "2004-11-28T12:00:00",
        "IssuedBy": "REPUBLIC OF KENYA",
        "additionalIdentityDetails": [
            {
                "documentNumber": "",
                "documentType": "",
                "issuedBy": ""
            }
        ],
        "address": {
            "provinceName": " ",
            "districtName": "",
            "locationName": "",
            "subLocationName": "",
            "villageName": ""
        }
    }
}
```


loans_credit_score(*customer: list*, *bureau: dict*, *loan: dict*) → dict

Get Loans and credit score.

Example Request Payload customer,bureau,loan


```
{
    "customer": [{
        "id": "",
        "fullName": "",
        "firstName": "",
        "lastName": "",
        "shortName": "",
        "title": "",
```


(continues on next page)

(continued from previous page)

```
        "mobileNumber": "",
        "dateOfBirth": "1999-01-31",
        "identityDocument": {
            "documentType": "NationalID",
            "documentNumber": "12365478"
        }
    }],
    "bureau": {
        "reportType": "Mobile",
        "countryCode": "KE"
    },
    "loan": {
        "amount": "5000"
    }
}
```

Example Response

```
{
    "Person": {
        "PersonName": {},
        "IdentityDocument": {
            "IdentityDocumentID": "1234568",
            "IdentityDocumentType": "National ID"
        }
    },
    "CreditAccountsSummary": [
        {
            "AccountIdentifier": {
                "AccountID": "0011547896523",
                "AccountCurrency": {}
            },
            "AccountType": "36",
            "AccountOpenDate": "17012014",
            "AccountOwnership": "true",
            "Balance": "0.00000",
            "DelinquencyStatus": "No delinquency",
            "Original_Amount": [
                "65000.00000",
                "65000.00000"
            ],
            "PastDueAmount": "0.00000",
            "LastPaymentAmount": "5000.00000",
            "LastPaymentReceivedDate": "20062014",
            "NoofDelayed_Payments": "0",
            "PostedDateTime": "30062014",
            "AccountStatus": "F",
            "LoanAccount": {
                "PastDueDate": {},
                "LoanHighestDaysInArrears": {}
            }
        },
```

(continues on next page)

(continued from previous page)


```
        {
            "AccountIdentifier": {
                "AccountID": "0011547896523",
                "AccountCurrency": {}
            },
            "AccountType": "09",
            "AccountOpenDate": "09062011",
            "AccountOwnership": "true",
            "Balance": "106458.00000",
            "DelinquencyStatus": "No delinquency",
            "Original_Amount": [
                "200000.00000",
                "200000.00000"
            ],
            "PastDueAmount": "0.00000",
            "LastPaymentAmount": "1667.00000",
            "LastPaymentReceivedDate": "15062018",
            "NoofDelayed_Payments": "0",
            "PostedDateTime": "30062018",
            "AccountStatus": "W",
            "LoanAccount": {
                "PastDueDate": {},
                "LoanHighestDaysInArrears": {}
            }
        },
        {
            "AccountIdentifier": {
                "AccountID": "0011547896523",
                "AccountCurrency": {}
            },
            "AccountType": "36",
            "AccountOpenDate": "14052014",
            "AccountOwnership": "true",
            "Balance": "0.00000",
            "DelinquencyStatus": "No delinquency",
            "Original_Amount": [
                "80000.00000",
                "80000.00000"
            ],
            "PastDueAmount": "0.00000",
            "LastPaymentAmount": "6960.00000",
            "LastPaymentReceivedDate": "15122014",
            "NoofDelayed_Payments": "0",
            "PostedDateTime": "31122014",
            "AccountStatus": "F",
            "LoanAccount": {
                "PastDueDate": {},
                "LoanHighestDaysInArrears": {}
            }
        },
        {
            "AccountIdentifier": {
```


(continues on next page)

(continued from previous page)


```
                "AccountID": "0011547896523",
                "AccountCurrency": {}
            },
            "AccountType": "36",
            "AccountOpenDate": "22092014",
            "AccountOwnership": "true",
            "Balance": "0.00000",
            "DelinquencyStatus": "No delinquency",
            "Original_Amount": [
                "1000.00000",
                "1000.00000"
            ],
            "PastDueAmount": "0.00000",
            "LastPaymentAmount": "1000.00000",
            "LastPaymentReceivedDate": "15102014",
            "NoofDelayed_Payments": "0",
            "PostedDateTime": "31102014",
            "AccountStatus": "F",
            "LoanAccount": {
                "PastDueDate": {},
                "LoanHighestDaysInArrears": {}
            }
        },
        {
            "AccountIdentifier": {
                "AccountID": "0011547896523",
                "AccountCurrency": {}
            },
            "AccountType": "36",
            "AccountOpenDate": "29122014",
            "AccountOwnership": "true",
            "Balance": "0.00000",
            "DelinquencyStatus": "No delinquency",
            "Original_Amount": [
                "80000.00000",
                "80000.00000"
            ],
            "PastDueAmount": "0.00000",
            "LastPaymentAmount": "6666.67000",
            "LastPaymentReceivedDate": "16032015",
            "NoofDelayed_Payments": "0",
            "PostedDateTime": "31032015",
            "AccountStatus": "F",
            "LoanAccount": {
                "PastDueDate": {},
                "LoanHighestDaysInArrears": {}
            }
        },
        {
            "AccountIdentifier": {
                "AccountID": "0011547896523",
                "AccountCurrency": {}
```


(continues on next page)

(continued from previous page)


```
            },
            "AccountType": "36",
            "AccountOpenDate": "20032015",
            "AccountOwnership": "true",
            "Balance": "0.00000",
            "DelinquencyStatus": "No delinquency",
            "Original_Amount": [
                "80000.00000",
                "80000.00000"
            ],
            "PastDueAmount": "0.00000",
            "LastPaymentAmount": "6666.67000",
            "LastPaymentReceivedDate": "16012016",
            "NoofDelayed_Payments": "0",
            "PostedDateTime": "31012016",
            "AccountStatus": "F",
            "LoanAccount": {
                "PastDueDate": {},
                "LoanHighestDaysInArrears": {}
            }
        },
        {
            "AccountIdentifier": {
                "AccountID": "N:000019:01:2015",
                "AccountCurrency": {}
            },
            "AccountType": "23",
            "AccountOpenDate": "06012015",
            "AccountOwnership": "false",
            "Balance": "0.00000",
            "DelinquencyStatus": "No delinquency",
            "Original_Amount": [
                "300000.00000",
                "300000.00000"
            ],
            "PastDueAmount": "0.00000",
            "LastPaymentAmount": "20562.00000",
            "LastPaymentReceivedDate": "27102017",
            "NoofDelayed_Payments": "0",
            "PostedDateTime": "31122017",
            "AccountStatus": "F",
            "LoanAccount": {
                "PastDueDate": {},
                "LoanHighestDaysInArrears": {}
            }
        },
        {
            "AccountIdentifier": {
                "AccountID": "068-P-12365478",
                "AccountCurrency": {}
            },
            "AccountType": "04",
```


(continues on next page)

(continued from previous page)


```
            "AccountOpenDate": "13102011",
            "AccountOwnership": "true",
            "Balance": "39844.00000",
            "DelinquencyStatus": "No delinquency",
            "Original_Amount": [
                "40000.00000",
                "40000.00000"
            ],
            "PastDueAmount": "0.00000",
            "LastPaymentAmount": "2500.00000",
            "LastPaymentReceivedDate": "16072018",
            "NoofDelayed_Payments": "0",
            "PostedDateTime": "31072018",
            "AccountStatus": "W",
            "LoanAccount": {
                "PastDueDate": {},
                "LoanHighestDaysInArrears": {}
            }
        },
        {
            "AccountIdentifier": {
                "AccountID": "068-P-25417854",
                "AccountCurrency": {}
            },
            "AccountType": "04",
            "AccountOpenDate": "19082015",
            "AccountOwnership": "true",
            "Balance": "0.00000",
            "DelinquencyStatus": "No delinquency",
            "Original_Amount": [
                "50000.00000",
                "50000.00000"
            ],
            "PastDueAmount": "0.00000",
            "LastPaymentAmount": "2500.00000",
            "LastPaymentReceivedDate": "13022018",
            "NoofDelayed_Payments": "0",
            "PostedDateTime": "31072018",
            "AccountStatus": "F",
            "LoanAccount": {
                "PastDueDate": {},
                "LoanHighestDaysInArrears": {}
            }
        },
        {
            "AccountIdentifier": {
                "AccountID": "0011547896523",
                "AccountCurrency": {}
            },
            "AccountType": "23",
            "AccountOpenDate": "02022016",
            "AccountOwnership": "true",
```


(continues on next page)

(continued from previous page)


```
            "Balance": "0.00000",
            "DelinquencyStatus": "No delinquency",
            "Original_Amount": [
                "80000.00000",
                "80000.00000"
            ],
            "PastDueAmount": "0.00000",
            "LastPaymentAmount": "6666.67000",
            "LastPaymentReceivedDate": "16122016",
            "NoofDelayed_Payments": "0",
            "PostedDateTime": "31012017",
            "AccountStatus": "F",
            "LoanAccount": {
                "PastDueDate": {},
                "LoanHighestDaysInArrears": {}
            }
        },
        {
            "AccountIdentifier": {
                "AccountID": "2569774",
                "AccountCurrency": {}
            },
            "AccountType": "12",
            "AccountOpenDate": "02062016",
            "AccountOwnership": "false",
            "Balance": "0.00000",
            "DelinquencyStatus": "No delinquency",
            "Original_Amount": [
                "3000.00000",
                "3000.00000"
            ],
            "PastDueAmount": "0.00000",
            "LastPaymentAmount": "3726.00000",
            "LastPaymentReceivedDate": "26122016",
            "NoofDelayed_Payments": "0",
            "PostedDateTime": "30062018",
            "AccountStatus": "F",
            "LoanAccount": {
                "PastDueDate": {},
                "LoanHighestDaysInArrears": {}
            }
        },
        {
            "AccountIdentifier": {
                "AccountID": "11110571749286",
                "AccountCurrency": {}
            },
            "AccountType": "23",
            "AccountOpenDate": "14022017",
            "AccountOwnership": "true",
            "Balance": "0.00000",
            "DelinquencyStatus": "No delinquency",
```


(continues on next page)

(continued from previous page)


```
            "Original_Amount": [
                "120000.00000",
                "120000.00000"
            ],
            "PastDueAmount": "0.00000",
            "LastPaymentAmount": "10000.00000",
            "LastPaymentReceivedDate": "15112017",
            "NoofDelayed_Payments": "0",
            "PostedDateTime": "31122017",
            "AccountStatus": "F",
            "LoanAccount": {
                "PastDueDate": {},
                "LoanHighestDaysInArrears": {}
            }
        },
        {
            "AccountIdentifier": {
                "AccountID": "JKCBDL1724301111",
                "AccountCurrency": {}
            },
            "AccountType": "12",
            "AccountOpenDate": "30082017",
            "AccountOwnership": "false",
            "Balance": "0.00000",
            "DelinquencyStatus": "No delinquency",
            "Original_Amount": [
                "5400.00000",
                "5400.00000"
            ],
            "PastDueAmount": "0.00000",
            "LastPaymentAmount": "None",
            "LastPaymentReceivedDate": "None",
            "NoofDelayed_Payments": "0",
            "PostedDateTime": "13122017",
            "AccountStatus": "A",
            "LoanAccount": {
                "PastDueDate": {},
                "LoanHighestDaysInArrears": {}
            }
        },
        {
            "AccountIdentifier": {
                "AccountID": "BCKMLD1802229762",
                "AccountCurrency": {}
            },
            "AccountType": "12",
            "AccountOpenDate": "08042018",
            "AccountOwnership": "false",
            "Balance": "0.00000",
            "DelinquencyStatus": "No delinquency",
            "Original_Amount": [
                "5400.00000",
```


(continues on next page)

(continued from previous page)


```
                "5400.00000"
            ],
            "PastDueAmount": "0.00000",
            "LastPaymentAmount": "None",
            "LastPaymentReceivedDate": "11062018",
            "NoofDelayed_Payments": "0",
            "PostedDateTime": "21062018",
            "AccountStatus": "A",
            "LoanAccount": {
                "PastDueDate": {},
                "LoanHighestDaysInArrears": {}
            }
        },
        {
            "AccountIdentifier": {
                "AccountID": "MKDLCB1814647289",
                "AccountCurrency": {}
            },
            "AccountType": "12",
            "AccountOpenDate": "26052018",
            "AccountOwnership": "false",
            "Balance": "5400.00000",
            "DelinquencyStatus": "No delinquency",
            "Original_Amount": [
                "5400.00000",
                "5400.00000"
            ],
            "PastDueAmount": "0.00000",
            "LastPaymentAmount": "5400.00000",
            "LastPaymentReceivedDate": "26052018",
            "NoofDelayed_Payments": "0",
            "PostedDateTime": "31052018",
            "AccountStatus": "W",
            "LoanAccount": {
                "PastDueDate": {},
                "LoanHighestDaysInArrears": {}
            }
        },
        {
            "AccountIdentifier": {
                "AccountID": "MKCDLB1818039369",
                "AccountCurrency": {}
            },
            "AccountType": "12",
            "AccountOpenDate": "29062018",
            "AccountOwnership": "false",
            "Balance": "2150.00000",
            "DelinquencyStatus": "No delinquency",
            "Original_Amount": [
                "2150.00000",
                "2150.00000"
            ],
```


(continues on next page)

(continued from previous page)


```
            "PastDueAmount": "0.00000",
            "LastPaymentAmount": "2150.00000",
            "LastPaymentReceivedDate": "29062018",
            "NoofDelayed_Payments": "0",
            "PostedDateTime": "30062018",
            "AccountStatus": "W",
            "LoanAccount": {
                "PastDueDate": {},
                "LoanHighestDaysInArrears": {}
            }
        },
        {
            "AccountIdentifier": {
                "AccountID": "MDLBBCK1821123688",
                "AccountCurrency": {}
            },
            "AccountType": "12",
            "AccountOpenDate": "29072018",
            "AccountOwnership": "false",
            "Balance": "2150.00000",
            "DelinquencyStatus": "No delinquency",
            "Original_Amount": [
                "2150.00000",
                "2150.00000"
            ],
            "PastDueAmount": "0.00000",
            "LastPaymentAmount": "2150.00000",
            "LastPaymentReceivedDate": "30072018",
            "NoofDelayed_Payments": "0",
            "PostedDateTime": "31072018",
            "AccountStatus": "W",
            "LoanAccount": {
                "PastDueDate": {},
                "LoanHighestDaysInArrears": {}
            }
        }
    ],
    "CreditBureau": {
        "score": "772",
        "creditApplications90Days": "0",
        "creditApplications180Days": "0",
        "creditApplications365Days": "0",
        "crbEnqiry90Days": "0",
        "crbEnqiry180Days": "0",
        "crbEnqiry365Days": "0",
        "BouncedCheques90Days": "0",
        "BouncedCheques180Days": "0",
        "BouncedCheques365Days": "0",
        "AcctNonPerformingCurrent": "0",
        "AcctNonPerformingHisto": "0",
        "AcctPerformingCurrent": "15",
        "AcctPerformingHisto": "NaN",
```


(continues on next page)

(continued from previous page)

```
        "IsFraud": "false",
        "isGuarantor": "false",
        "delinquency_code": "No delinquency"
    }
}
```

purchase_airtime(*customer: dict*, *airtime: dict*) → dict
Purchase airtime.

This gives an application the ability to purchase airtime from any telco in East and Central Africa. Example Request

Customer:

```
{
    "countryCode": "KE",
    "mobileNumber": "0765555131"
}
```

countryCode: the telco's ISO country code

mobileNumber: the mobile number you are purchasing airtime for

Airtime:

```
{
    "amount": "100",
    "reference": "692194625798",
    "telco": "Equitel"
}
```

telco the telco/provider. For example: Equitel, Safaricom , Airtel.

reference your transaction references. Should always be a 12 digit string

amount the airtime amount string

Example Response

```
{
    "referenceNumber": "4568899373748",
    "status": "SUCCESS"
}
```

receive_money_bill_payment(*biller: jengahq.receive_money.Biller*, *bill: jengahq.receive_money.Bill*, *payer: jengahq.receive_money.Payer*, *partnerId: str*, *remarks: str*)
Receive Money via bill Payments.

receive_money_bill_validation(*billerCode*, *customerRefNumber*, *amount*, *amountCurrency*)
Perform bill validation.

receive_money_eazzypay_push(*customer: jengahq.receive_money.Customer*, *transaction: jengahq.receive_money.Transaction*)
Receive Money from customer via Eazzy pay push.

receive_money_merchant_payment(*merchant:* jengahq.receive_money.Merchant, *payment:* jengahq.receive_money.Payment, *partner:* jengahq.receive_money.Partner)
Receive money via merchant payments.

receive_money_refund_payment(*customer:* jengahq.receive_money.Customer, *transaction:* jengahq.receive_money.RefundReverseTransaction)
Refund Payment from customer.

send_money_to_eft(*source:* jengahq.send_money.Source, *destination:* jengahq.send_money.EFTDest, *transfer:* jengahq.send_money.EFTTransfer)
Send money to EFT from equity bank.

send_money_to_equity(*source:* jengahq.send_money.Source, *destination:* jengahq.send_money.Dest, *transfer:* jengahq.send_money.Dest)
Send money to bank account within equity bank.

send_money_to_mobile_wallet(*source:* jengahq.send_money.Source, *destination:* jengahq.send_money.MobileDest, *transfer:* jengahq.send_money.MobileTransfer)
Send money to mobile wallets from equity bank.

send_money_to_pesalink_bank(*source:* jengahq.send_money.Source, *destination:* jengahq.send_money.PesalinkDest, *transfer:* jengahq.send_money.PesalinkTransfer)
Send money to Pesalink bank from equity bank.

send_money_to_pesalink_mobile(*source:* jengahq.send_money.Source, *destination:* jengahq.send_money.PesalinkMobileDest, *transfer:* jengahq.send_money.PesalinkTransfer)
Send money to Pesalink bank from equity bank.

send_money_to_rtgs(*source:* jengahq.send_money.Source, *destination:* jengahq.send_money.RTGSDest, *transfer:* jengahq.send_money.Transfer)
Send money to RTGS from equity bank.

send_money_to_swift(*source:* jengahq.send_money.Source, *destination:* jengahq.send_money.SWIFTDest, *transfer:* jengahq.send_money.SWIFTTransfer)
Send money to SWIFT from equity bank.

signature(*request_hash_fields: tuple*)
Return Signature to be used in request header.

Build a String of concatenated values of the request fields with following order: as specificied by the API endpoint The resulting text is then signed with Private Key and Base64 encoded.

Takes a tuple of request fields in the order that they should be concatenated, hashes them with SHA-256,signs the resulting hash and returns a Base64 encoded string of the resulting signature

jengahq.auth.**generate_key_pair**()
Generate a public/private key pair.

Generates a Public/Public RSA Key Pair which is store in the current User's **HOME** directory under the **.JengaAPI/keys/** Directory

2.2 jengahq.helpers

JengaHQ helpers.

jengahq.helpers.**timenow**()
Return now date.

jengahq.helpers.**todaystr**()
Return today date as string %Y-%m-%d.

jengahq.helpers.**token_expired**(*last_auth*)
Return true if last_auth is less than 3s from now.

2.3 jengahq.send_money

Send Money Module.

class jengahq.send_money.**Dest**(*account_number*, *name*, *country_code='KE'*, *type='bank'*)
Bases: object

Destination for send money.

to_json()
Convert to json.

class jengahq.send_money.**EFT**(*source:* jengahq.send_money.Source, *dest:* jengahq.send_money.EFTDest, *transfer:* jengahq.send_money.Transfer)
Bases: *jengahq.send_money.IFT*

EFT Funds Transfer.

property sigkey
Return text to generate signature.

class jengahq.send_money.**EFTDest**(*account_number*, *name*, *bankCode*, *branchCode*, *country_code='KE'*)
Bases: *jengahq.send_money.Dest*

EFT Destination.

to_json()
Convert to json.

class jengahq.send_money.**EFTTransfer**(*amount*, *reference*, *currencyCode*, *date*, *description*)
Bases: *jengahq.send_money.Transfer*

EFT Transfer.

class jengahq.send_money.**IFT**(*source:* jengahq.send_money.Source, *dest:* jengahq.send_money.Dest, *transfer:* jengahq.send_money.Transfer)
Bases: object

Within Equity bank Funds Tranfer.

property body_payload
Return Body Payload.

property sigkey
Return text to generate signature.

class jengahq.send_money.**IFTMobile**(*source:* jengahq.send_money.Source, *dest:* jengahq.send_money.MobileDest, *transfer:* jengahq.send_money.Transfer)

Bases: *jengahq.send_money.IFT*

Within Equity to mobile funds transfer.

property sigkey
Return text to generate signature.

class jengahq.send_money.**MobileDest**(*mobile_number*, *name*, *walletName='Mpesa'*, *country_code='KE'*)

Bases: *jengahq.send_money.Dest*

Mobile Destination.

to_json()
Convert to json.

class jengahq.send_money.**MobileTransfer**(*amount*, *reference*, *currencyCode*, *date*, *description*)

Bases: *jengahq.send_money.Transfer*

Mobile Transfer.

class jengahq.send_money.**Pesalink**(*source:* jengahq.send_money.Source, *dest:* jengahq.send_money.Dest, *transfer:* jengahq.send_money.Transfer)

Bases: *jengahq.send_money.IFT*

Pesalink Funds Transfer.

property sigkey
Return text to generate signature.

class jengahq.send_money.**PesalinkDest**(*account_number*, *mobile_number*, *name*, *bankCode*, *country_code='KE'*)

Bases: *jengahq.send_money.Dest*

Pesa Link Bank Account Destination.

to_json()
Convert to json.

class jengahq.send_money.**PesalinkMobileDest**(*mobile_number*, *name*, *bankCode*, *country_code='KE'*)

Bases: *jengahq.send_money.Dest*

PesalinkMobile Destination.

to_json()
Convert to json.

class jengahq.send_money.**PesalinkTransfer**(*amount*, *reference*, *currencyCode*, *date*, *description*)

Bases: *jengahq.send_money.Transfer*

PesaLink Mobile Transfer.

class jengahq.send_money.**RTGS**(*source:* jengahq.send_money.Source, *dest:* jengahq.send_money.Dest, *transfer:* jengahq.send_money.Transfer)

Bases: *jengahq.send_money.IFT*

RTGS Funds Transfer.

property sigkey
Return text to generate signature.

class jengahq.send_money.**RTGSDest**(*account_number*, *name*, *bankCode*, *country_code='KE'*)
Bases: *jengahq.send_money.Dest*

RTGS Destination.

to_json()
Convert to json.

class jengahq.send_money.**SWIFT**(*source:* jengahq.send_money.Source, *dest:* jengahq.send_money.Dest, *transfer:* jengahq.send_money.Transfer)
Bases: *jengahq.send_money.RTGS*

SWIFT Funds Transfer.

class jengahq.send_money.**SWIFTDest**(*account_number*, *name*, *bankBic*, *addressline1*, *country_code='KE'*)
Bases: *jengahq.send_money.Dest*

SWIFT Destination.

to_json()
Convert to json.

class jengahq.send_money.**SWIFTTransfer**(*amount*, *reference*, *currencyCode*, *date*, *description*, *chargeOption='OTHER'*, *type='SWIFT'*)
Bases: *jengahq.send_money.Transfer*

SWIFT Transfer.

to_json()
Convert to json.

class jengahq.send_money.**Source**(*account_number*, *name*, *country_code='KE'*)
Bases: `object`

Funds Transfer Source.

to_json()
Convert to json.

class jengahq.send_money.**Transfer**(*amount*, *reference*, *currencyCode*, *date*, *description*, *type='InternalFundsTransfer'*)
Bases: `object`

Funds Transfer.

to_json()
Convert to json.

jengahq.send_money.**transfer_payload**(*src:* jengahq.send_money.Source, *dst:* jengahq.send_money.Dest, *transfer:* jengahq.send_money.Transfer) → dict
Return internal funds transfer payload.

2.4 jengahq.receive_money

Receive money module.

class jengahq.receive_money.**Bill**(*amount*, *currency*, *reference*)
Bases: object

Bill class.

to_json()
Return bill as json.

class jengahq.receive_money.**BillPayment**(*biller:* jengahq.receive_money.Biller, *bill:* jengahq.receive_money.Bill, *payer:* jengahq.receive_money.Payer, *partnerId*, *remarks*)
Bases: object

Bill Payments.

property body_payload
Return body payload as json.

property sigkey
Return tuple of fields to gen signature.

class jengahq.receive_money.**BillValidation**(*billerCode*, *customerRefNumber*, *amount*, *amountCurrency*)
Bases: object

Bill Validation.

property body_payload
Return body payload as json.

property sigkey
Return Tuple to gen signature.

class jengahq.receive_money.**Biller**(*billerCode*, *countryCode='KE'*)
Bases: object

Biller class.

to_json()
Return biller as json.

class jengahq.receive_money.**Customer**(*mobileNumber*, *countryCode='KE'*)
Bases: object

Customer class.

to_json()
Return customer as json.

class jengahq.receive_money.**EazzypayPush**(*customer:* jengahq.receive_money.Customer, *transaction:* jengahq.receive_money.Transaction, *merchantCode*)
Bases: object

Easy Pay Push.

property body_payload
Return body payload as json.

property sigkey
Return tuple of fields to gen signature.

class jengahq.receive_money.**Merchant**(*till*)
Bases: object

Merchant class.

to_json()
Return merchant as json.

class jengahq.receive_money.**MerchantPayment**(*merchant:* jengahq.receive_money.Merchant, *payment:* jengahq.receive_money.Payment, *partner:* jengahq.receive_money.Partner)
Bases: object

Merchant Payments.

property body_payload
Return body payload as json.

property sigkey
Return tuple of fields to gen signature.

class jengahq.receive_money.**Partner**(*id*, *reference*)
Bases: object

Partner class.

to_json()
Return partner as json.

class jengahq.receive_money.**Payer**(*name*, *account*, *reference*, *mobileNumber*)
Bases: object

Payer class.

to_json()
Return payer as json.

class jengahq.receive_money.**Payment**(*amount*, *currency*, *reference*)
Bases: object

Payment class.

to_json()
Return payment as json.

class jengahq.receive_money.**RefundReversePayment**(*customer:* jengahq.receive_money.Customer, *transaction:* jengahq.receive_money.RefundReverseTransaction)
Bases: object

Refund Reverse Payment.

property body_payload
Return body payload as json.

property sigkey
Return tuple of fields to gen signature.

class jengahq.receive_money.**RefundReverseTransaction**(*amount*, *description*, *reference*, *service='EazzyPayOnline'*, *channel='EAZ'*, *type='refund'*)
Bases: *jengahq.receive_money.Transaction*

Refund or reverse Transaction class.

to_json()
Return transaction as json.

class jengahq.receive_money.**Transaction**(*amount*, *description*, *reference*, *type='EazzyPayOnline'*)
Bases: object

Transaction class.

to_json()
Return transaction as json.

2.5 Errors and Exceptions

Downstream Provider Error responses.

error	900101	Invalid merchant Cipher Text
error	900101	Invalid merchant certificate
error	900101	invalid merchant public key configuration – is empty
error	900101	Invalid merchant public key configuration —Public key is null
error	900101	Invalid merchant code provided

Merchant Authorization:

Status	Code	Response Message
error	401101	Unauthorized
error	401102	Service Not available
error	401103	Service Not available

Merchant Account/ Profile Validation:

error	104101	Validation of account failed
error	104102	Merchant daily transaction limit exceeded
error	104103	Insufficient Funds
error	104104	Invalid charge configuration
error	104105	Server Error
error	104106	Server error - could not save to database
error	104107	Invalid merchant configuration - No billing account found

RTGS:

error	118102	Service Not Available
error	118108	The source account has insufficient balance amount
error	103101	Invalid Account Type.
error	103102	Invalid account number .
error	103104	Insufficient funds.
error	103105	Transfer Limit Exceeded .
error	103106	Request Failed, please contact Bank.
error	103107	Invalid Currency/Transaction Amount.
error	103108	System Failure please retry .
error	103109	System malfunction .

Send Money - Within Equity Bank and Equitel

error	400103	Failed, invalid destination bank account
error	400104	Failed, insufficient funds
error	400105	Failed, general error
error	400106	Failed, destination bank error.
error	400107	Failed, destination bank error. Cut-over in progress
error	400108	Failed, destination bank error. System malfunction
error	400109	Failed, destination bank error. Duplicate transmission
error	400111	Failed, missing parameter in integration to gateway system
error	400112	Failed, could not process this request. System failure.
error	400101	Duplicate Transaction/ Payment Reference
error	400113	Failed, sender daily limit exceeded
error	103120	Failed, banking service could not process the request
error	400114	Failed, unable to process request. Third party is unavailable
error	400115	Failed, unhandled error
error	400116	Failed, 12 digit transaction reference required
error	118110	Failed, maximum transfer amount is 999,999 per transaction

Pesalink

error	100116	Failed, pesalink or destination bank error
error	100124	Failed, invalid amount
error	100128	Failed, pesalink or destination bank error
error	100130	Failed, pesalink or destination bank error
error	100134	Failed, amount less than minimum allowed
error	100207	Failed, destination cannot be found for routing
error	100209	Failed, pesalink or destination bank error
error	100210	Failed, transaction timed out
error	100211	Failed, pesalink or destination bank error
error	100232	Failed, banking service could not process the request
error	400101	Duplicate Transaction/ Payment Reference
error	100222	Failed, suspected fraud
error	400102	Failed, invalid destination bank account
error	400103	Failed, invalid destination bank account
error	400104	Failed, insufficient funds
error	400105	Failed, general error
error	400106	Failed, destination bank error.
error	400107	Failed, destination bank error. Cut-over in progress
error	400108	Failed, destination bank error. System malfunction
error	400109	Failed, destination bank error. Duplicate transmission
error	400110	Failed, bank not on pesalink
error	400111	Failed, missing parameter in integration to gateway system
error	400112	Failed, could not process this request. System failure.
error	400113	Failed, sender daily limit exceeded
error	400114	Failed, unable to process request. Third party is unavailable
error	400115	Failed, unhandled error
error	400116	Failed, 12 digit transaction reference required

Purchase Airtime

error	107101	Service Not available
error	107102	Invalid Provider
error	107103	Invalid Parameters
error	107104	Invalid Mobile Number
error	107105	Invalid Service Type
error	107106	Invalid Amount
error	107107	Payment reference Number should be 12 digits
error	107108	Authentication Failed
error	107109	The maximum amount allowed for transfers per transaction is KES 8000
error	107110	Failed. Transaction has been reversed to your Account {account}. Ref.{reference}.
error	107111	Dear Customer, we are unable to complete your request at this time. Kindly try again later. Reference Number {ref}
error	107112	Failed.Transaction Ref.#RefNo# unsuccessful. Kindly call 100 for assistance
error	400101	Duplicate Transaction/ Payment Reference

Send Money - To Mobile Wallet (Airtel and M-PESA)

error	400101	Duplicate Transaction/ Payment Reference
error	400104	Failed, insufficient funds
error	400105	Failed, general error
error	400112	Failed, could not process this request. System failure.
error	400113	Failed, sender daily limit exceeded
error	400114	Failed, unable to process request. Third party is unavailable
error	400115	Failed, unhandled error
error	400116	Failed, 12 digit transaction reference required
error	103120	Failed, banking service could not process the request
error	105154	Failed, we cannot process your request at the moment due to system failure at Airtel
error	105156	Failed, the number is not registered for Airtel money
error	105157	Failed, credit party customer type can't be supported by the service
error	105158	Failed, Maximum transfer amount is 70000 per transaction
error	105160	Failed, invalid mobile wallet
error	118101	Customer not registered for Equitel Mobile Banking

Bill and Till Payments

error	102101	Missing parameter
error	102102	invalid Reference Number
error	102103	Invalid Biller Code/Till Number
error	102104	Invalid Bill Reference
error	102105	Reversal Failed
error	102106	Insufficient Balance, Invalid Account
error	102107	System Failure please retry
error	102108	Reversal Failed
error	102110	Service Not available
error	400101	Duplicate Transaction/ Payment Reference

Eazzypay Push

error	114101	Merchant not registered for service
error	114102	Amount is below minimum that is allowed for transaction.
error	114103	The maximum amount limit is reached
error	114104	Customer is not registered
error	114105	Maximum daily amount per-day limit is reached for this payment
error	114106	Maximum transactions per-day limit is reached for this payment
error	114107	Invalid parameters. Kindly check with bank team
error	400101	Duplicate Transaction/ Payment Reference

Lipa na M-Pesa Online

error	117101	Bad Request - Invalid TransactionType
error	117101	Bad Request - Invalid Password
error	117101	Bad Request - Invalid PartyB
error	117101	Bad Request - Invalid CallBackURL
error	117101	Bad Request - Invalid Remarks
error	117101	Bad Request - Invalid PartyA
error	117101	Bad Request - Invalid Amount
error	117102	[MerchantValidate] - Wrong credentials
error	117102	Request failed: Output data invalid
error	117104	Error Occured: Spike Arrest Violation
error	500101	Generic error. Please contact our support.

Get Payment Status

error	111101	Failed. Invalid input parameters
error	111102	Bad request - Transaction not found.

Refund Payment

error	113101	Service Not available
error	113102	Invalid request format
error	113103	Invalid Transaction Status code for Reverse the Transaction
error	113104	Service Not available
error	113105	Due to technical problem we are not able to process your request

Identity Verification

error	115101	ID number used for search contains invalid symbols
error	115102	Serial number used for search contains invalid symbols
error	115103	At least one of search parameters should be filled
error	115104	Search can't be done if one of mandatory fields is empty. All input parameters should be filled
error	115105	There is no record found in IPRS corresponding search parameters
error	115106	Service Not available
error	115107	Service Not available
error	115108	Service Not available
error	115109	Service Not available
error	115110	Service Not available
error	115111	Passport number contains illegal symbols
error	115112	At least one of search parameters should be filled
error	115113	Search can't be done if one of mandatory fields is empty. All input parameters should be filled
error	115114	User can't invoke verification methods because he didn't pass the authorization or don't have sufficient privileges
error	115115	Service Not available
error	115116	The image sent doesn't correspond to the fingerprint format
error	115117	Service Not available
error	115118	Service Not available
error	115119	Service Not available

Account Balance

error	8504	No record could be retrieved

Create Bill

error	106101	Unauthorized
error	109101	invalid charge configuration
error	109102	invalid channel code
error	109103	Bill already cleared
error	109104	Invalid service configuration
error	109105	Invalid charge channel configuration code
error	109106	Invalid merchant Outlet code specified
error	109107	Invalid merchant code
error	109108	Invalid request data. Please check your payload
error	109109	Service not available

Credit & Debit Card

error	101101	Transaction could not be processed
error	101102	Transaction declined – contact issuing bank
error	101103	No reply from Processing Host
error	101104	Card has expired
error	101105	Insufficient credit
error	101106	Error Communicating with Bank
error	101107	Message Detail Error (Invalid PAN, Invalid Expiry Date)
error	101108	Transaction declined –transaction type not supported
error	101109	Bank Declined Transaction – Do Not Contact Bank
error	116101	Payment Not Updated
error	109101	invalid charge configuration
error	109102	invalid channel code
error	109103	Bill already cleared
error	109104	Invalid service configuration
error	109105	Invalid charge channel configuration code
error	109106	Invalid merchant Outlet code specified
error	109107	Invalid merchant code
error	109108	Invalid request data. Please check your payload
error	109109	Service not available

Receive Payments - Mobile Wallets

error	114101	Merchant not registered for service
error	114102	Amount is below minimum that is allowed for transaction.
error	114103	The maximum amount limit is reached
error	114104	Customer is not registered
error	114105	Maximum daily amount per-day limit is reached for this payment
error	114106	Maximum transactions per-day limit is reached for this payment
error	114107	Invalid parameters. Kindly check with bank team
error	111101	Failed. Invalid input parameters.
error	111102	Bad request - Transaction not found
error	116101	Payment Not Updated

Query Payment

error	112101	Amount paid is less or more than bill amount
error	112102	No transaction found for order ref
error	112103	Bill not found
error	112104	Invalid merchant code
error	112105	invalid service map configuration
error	112106	Due to technical problem we are not able to process your request
error	112107	Invalid service configuration
error	112108	Bill already cleared

Query Bill

error	110101	Invalid merchant code
error	110102	Invalid request data. Please check your payload
error	110103	bill doesn't exist

Generic Error

```
{
"response_status": "error",
"response_code": "500101",
"response_msg": "Generic error. Please contact our support."
}
```

Target Error

200 OK

```
{
"response_status": "error",
"response_code": "{$.response code}",
"response_msg": "{$.message}"
}
```

jengahq.exceptions.**generate_reference**() → str
Generate reference unique 12 digit string.

Generate a transaction reference Should always be a 12 digit String

jengahq.exceptions.**handle_response**(*response*)
Handle response.

Handles Responses From the JengaHQ API and Raises Exceptions appropriately as errors occur and returns a *dict* object from the *json* response

CHAPTER

THREE

INDICES AND TABLES

- genindex
- modindex
- search

PYTHON MODULE INDEX

j

INDEX

L

M

P

R

S

T