

---

# equity-jenga-api

May 30, 2020



---

## Contents:

---

<b>1</b>	<b>Getting Started</b>	<b>1</b>
1.1	Installation	1
1.2	First Things First	1
1.2.1	Generating an openSSL private/public key pair.	1
1.3	Using The APIs	2
1.3.1	Creating JengaRequest objects	2
<b>2</b>	<b>Equity_Jenga Package</b>	<b>3</b>
2.1	Jenga Submodules	3
2.2	equity_jenga.api.auth	3
2.3	equity_jenga.api.account_services.balances	4
2.4	equity_jenga.api.account_services.statements	5
2.5	equity_jenga.api.account_services.inquiries	7
2.6	equity_jenga.api.airtime	8
2.7	equity_jenga.api.loans	9
2.8	equity_jenga.api.forex	18
2.9	equity_jenga.api.kyc	18
2.10	equity_jenga.api.exceptions	20
<b>3</b>	<b>Indices and tables</b>	<b>29</b>
	<b>Python Module Index</b>	<b>31</b>
	<b>Index</b>	<b>33</b>



### 1.1 Installation

```
pip install equity-jenga-api
```

### 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 JengaRequest objects

By Design all API classes provided inherit from the `equity_jenga.api.auth.JengaAuth` class which takes the following parameters:

```
class equity_jenga.api.auth.JengaAuth (authorization_token: str, merchant_code: str, env: str, private_key='~/JengaAPI/keys/privatekey.pem', sandbox_url='https://sandbox.jengahq.io', live_url="")
```

Jenga Base Authentication Class

#### Params

:authorization\_token:: the bearer token used to access the API, i.e apiKey  
 :merchant\_code:: the merchant code provided by JengaHQ  
 :private\_key:: the path to the merchant private key default is “~/JengaAPI/keys/privatekey.pem”  
 :env:: the environment in which the API is to be used either *sandbox* or *production*  
 :sandbox\_url:: the url used to access the Sandbox API  
 :live\_url:: the url used to access the Production API

#### Example

```
from equity_jenga import api
jengabase = api.auth.JengaApi(
authorization_token="Bearer TofFGUeU9y448idLCKVAe35LmAtL",
merchant_code="4144142283",
env="sandbox",
)
# Similarly for all classes Inheriting from JengaAuth
acc_Bal = api.auth.account_services.balances.AccountBalance(
authorization_token="Bearer TofFGUeU9y448idLCKVAe35LmAtL",
merchant_code="4144142283",
env="sandbox",
)
# Get Available Balances
available_bal=acc_Bal.available(countryCode='KE',accountId="0011547896523")
print(available_bal)
# Opening and Closing Balances
opening_and_closing_bal=acc_Bal.get_opening_and_closing(accountId="0011547896523
↪"countryCode='KE',date="2020-03-21")
print(opening_and_closing_bal)
```

#### signature (request\_hash\_fields: tuple)

Build a String of concatenated values of the request fields with following order: as specified 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

## 2.1 Jenga Submodules

## 2.2 equity\_jenga.api.auth

```
class equity_jenga.api.auth.JengaAuth (authorization_token: str, mer-  
chant_code: str, env: str, private_key='/home/docs/.JengaApi/keys/privatekey.pem',  
sandbox_url='https://sandbox.jengahq.io',  
live_url="")
```

Bases: object

Jenga Base Authentication Class

### Params

*:authorization\_token:*: the bearer token used to access the API, i.e apiKey  
*:merchant\_code:*: the merchant code provided by JengaHQ  
*:private\_key:*: the path to the merchant private key default is “~/JengaAPI/keys/privatekey.pem”  
*:env:*: the environment in which the API is to be used either *sandbox* or *production*  
*:sandbox\_url:*: the url used to access the Sandbox API  
*:live\_url:*: the url used to access the Production API

### Example

```
from equity_jenga import api  
jengabase = api.auth.JengaApi(  
authorization_token="Bearer TofFGUeU9y448idLCKVAe35LmAtL",  
merchant_code="4144142283",  
env="sandbox",  
)  
# Similarly for all classes Inheriting from JengaAuth  
acc_Bal = api.auth.account_services.balances.AccountBalance(  
authorization_token="Bearer TofFGUeU9y448idLCKVAe35LmAtL",  
merchant_code="4144142283",  
env="sandbox",
```

(continues on next page)

(continued from previous page)

```

)
# Get Available Balances
available_bal=acc_Bal.available(countryCode='KE',accountId="0011547896523")
print(available_bal)
# Opening and Closing Balances
opening_and_closing_bal=acc_Bal.get_opening_and_closing(accountId="0011547896523
↪"countryCode='KE',date="2020-03-21")
print(opening_and_closing_bal)

```

**signature** (*request\_hash\_fields: tuple*)

Build a String of concatenated values of the request fields with following order: as specified 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

`equity_jenga.api.auth.generate_key_pair()`

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

## 2.3 equity\_jenga.api.account\_services.balances

**class** `equity_jenga.api.account_services.balances.AccountBalance` (*authorization\_token:*

*str,* *mer-*

*chant\_code:*

*str,* *env:*

*str,* *pri-*

*vate\_key='/home/docs/.JengaApi/keys/p*

*sand-*

*box\_url='https://sandbox.jengahq.io',*

*live\_url='')*

Bases: `auth.JengaAuth`

**authenticate** (*countryCode, accountId*) → dict

**available** (*countryCode, accountId*) → dict

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\_opening\_and\_closing** (*accountId, countryCode, date*)

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"
    }
  ]
}
```

## 2.4 equity\_jenga.api.account\_services.statements

```
class equity_jenga.api.account_services.statements.FullStatement (authorization_token:
                                                                    str, merchant_code:
                                                                    str, env:
                                                                    str, private_key='~/home/docs/.JengaApi/keys/
                                                                    sandbox_url='https://sandbox.jengahq.io'
                                                                    live_url='')
```

Bases: `auth.JengaAuth`

**authenticate** (*accountNumber*, *countryCode*, *toDate*)

**retrieve** (*payload*)

Example Payload

```
{
  "countryCode": "KE",
  "accountNumber": "0011547896523",
  "fromDate": "2018-01-18",
  "toDate": "2018-04-19",
  "limit": 20,
  "reference": "",
  "serial": "",
  "postedDateTime": "",
  "date": "",
  "runningBalance": {
    "currency": "",
    "amount": 0.0
  }
}
```

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
↔",
      "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
      }
    }
  ]
}

```

```

class equity_jenga.api.account_services.statements.MiniStatement (authorization_token:
    str, merchant_code:
    str, env:
    str, private_key='home/docs/.JengaApi/keys
    sand-
    box_url='https://sandbox.jengahq.io'
    live_url='')

```

Bases: `auth.JengaAuth`

**authenticate** (*countryCode*, *accountNumber*)

**retrieve** (*countryCode*, *accountNumber*)

Example Response

```
{
  "accountNumber": "0011547896523",
  "currency": "KES",
  "balance": 1000,
  "transactions": [
    {
      "chequeNumber": null,
      "date": "2017-01-01T00:00:00",
      "description": "EAZZY-FUNDS TRNSF TO A/C XXXXXXXXXXXXX",
      "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",
      "amount": "220",
      "type": "Debit"
    },
    {
      "chequeNumber": null,
      "date": "2017-01-05T00:00:00",
      "description": "SI ACCOUNT TO ACCOUNT THIRD PA",
      "amount": "20",
      "type": "Debit"
    }
  ]
}
```

## 2.5 equity\_jenga.api.account\_services.inquiries

```
class equity_jenga.api.account_services.inquiries.AccountInquiry (authorization_token:
    str, merchant_code:
    str, env:
    str, private_key='/home/docs/.JengaApi/keys/
    sand-
    box_url='https://sandbox.jengahq.io'
    live_url='')
```

Bases: `auth.JengaAuth`

**account\_details** (*countryCode*, *accountNumber*)

Example Response

```
{
  "account": {
    "number": "0011547896523",
    "branchCode": "017",
    "currency": "KES",
    "status": "Active"
  },
  "customer": [
    {
      "id": "100200300",
      "name": "A N.Other",
      "type": "Retail"
    }
  ]
}
```

**authenticate** (*countryCode*, *accountNumber*)

## 2.6 equity\_jenga.api.airtime

```
class equity_jenga.api.airtime.Airtime (authorization_token: str, merchant_code: str, env: str, private_key='/home/docs/.JengaApi/keys/privatekey.pem', sandbox_url='https://sandbox.jengahq.io', live_url="")
```

Bases: *equity\_jenga.api.auth.JengaAuth*

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

**authenticate** (*merchantCode*: str, *airtimeTelco*: str, *airtimeAmount*: str, *airtimeReference*: str) → dict

**purchase\_airtime** (*customer*: dict, *airtime*: dict) → dict

Example Request :*countryCode*:: the telco's ISO country code :*mobileNumber*:: the mobile number you are purchasing airtime for

:*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

```
{
  "customer": {
    "countryCode": "KE",
    "mobileNumber": "0765555131"
  },
  "airtime": {
    "amount": "100",
    "reference": "692194625798",
    "telco": "Equitel"
  }
}
```

Example Response

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

## 2.7 equity\_jenga.api.loans

```
class equity_jenga.api.loans.CreditScore (authorization_token: str, merchant_code: str, env: str, private_key='/home/docs/.JengaApi/keys/privatekey.pem', sandbox_url='https://sandbox.jengahq.io', live_url='')
```

Bases: *equity\_jenga.api.auth.JengaAuth*

This API (first of its kind) aggregates all the Credit Reference Bureaus in East Africa as well as Equity Bank's own internal scoring to give a response on the credit worthiness (or not) of an individual. No need to perform countless integrations. This web service does it for you.

**authenticate** (*dateOfBirth: str, merchantCode: str, documentNumber: str*) → dict

**credit\_score** (*customer: list, bureau: dict, loan: dict*) → dict

Example Request Payload customer,bureau,loan

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

(continues on next page)

```
    },
    "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": {}
        }
      },
      {
        "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": {}
        }
      }
    ]
  }
}
```

(continues on next page)

(continued from previous page)

```

    },
    "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": {
        "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": [

```

(continues on next page)

(continued from previous page)

```

        "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": {}
    },
    "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",

```

(continues on next page)

(continued from previous page)

```

    "PostedDateTime": "31122017",
    "AccountStatus": "F",
    "LoanAccount": {
      "PastDueDate": {},
      "LoanHighestDaysInArrears": {}
    }
  },
  {
    "AccountIdentifier": {
      "AccountID": "068-P-12365478",
      "AccountCurrency": {}
    },
    "AccountType": "04",
    "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": {}
    }
  },

```

(continues on next page)

(continued from previous page)

```

{
  "AccountIdentifier": {
    "AccountID": "0011547896523",
    "AccountCurrency": {}
  },
  "AccountType": "23",
  "AccountOpenDate": "02022016",
  "AccountOwnership": "true",
  "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",

```

(continues on next page)

(continued from previous page)

```

    "AccountOwnership": "true",
    "Balance": "0.00000",
    "DelinquencyStatus": "No delinquency",
    "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",
      "5400.00000"
    ],
  },

```

(continues on next page)

(continued from previous page)

```
"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"
  ],
  "PastDueAmount": "0.00000",
  "LastPaymentAmount": "2150.00000",
  "LastPaymentReceivedDate": "29062018",
  "NoofDelayed_Payments": "0",
  "PostedDateTime": "30062018",
  "AccountStatus": "W",
  "LoanAccount": {
```

(continues on next page)

(continued from previous page)

```

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

```

## 2.8 equity\_jenga.api.forex

```
class equity_jenga.api.forex.Forex (authorization_token: str, merchant_code: str, env: str,
                                     private_key='/home/docs/.JengaApi/keys/privatekey.pem',
                                     sandbox_url='https://sandbox.jengahq.io', live_url='')
```

Bases: `equity_jenga.api.auth.JengaAuth`

The Foreign Exchange Rates API Provides Easy Access To The Equity Bank Daily Currency Conversion Rate For Major Currencies

**authenticate** () → dict

**forex\_rates** (countryCode: str, currencyCode: str) → dict

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"
}
```

## 2.9 equity\_jenga.api.kyc

This web service enables your application to query the various registrar of persons in the various countries in East Africa.

Currently available for Kenya and Rwanda only.

<https://sandbox.jengahq.io/customer-test/v2/identity/verify>

```
class equity_jenga.api.kyc.IDSearchVerification (authorization_token: str, merchant_code: str, env: str, private_key='/home/docs/.JengaApi/keys/privatekey.pem',
                                                  sandbox_url='https://sandbox.jengahq.io', live_url='')
```

Bases: `equity_jenga.api.auth.JengaAuth`

**authenticate** (merchantCode, documentNumber, countryCode)

Params

**MerchantCode** Your MerchantCode

**DocumentNumber** Document Number to Verify

**CountryCode** The Country Code

**Returns****Headers****Authorization** string the bearer token used to access the API**Content-Type** string the content type of the payload**Signature** string A SHA-256 signature to proof that this request is coming from the merchant. Build a String of concatenated values of the request fields with the following order: merchantcode documentNumber countryCode. The resulting text is then signed with Private Key and Base64 encoded.

Should return the headers to pass to the particular endpoint

**verify** (*identity: dict*)

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": "",

```

(continues on next page)

(continued from previous page)

```

        "issuedBy": ""
      }
    ],
    "address": {
      "provinceName": " ",
      "districtName": "",
      "locationName": "",
      "subLocationName": "",
      "villageName": ""
    }
  }
}

```

## 2.10 equity\_jenga.api.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

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

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

`equity_jenga.api.exceptions.generate_reference ()` → str  
Generate a transaction reference Should always be a 12 digit String

`equity_jenga.api.exceptions.handle_response (response)`  
Handles Responses From the JengaHQ API and Raises Exceptions appropriately as errors occur and returns a *dict* object from the *json* response



## CHAPTER 3

---

### Indices and tables

---

- `genindex`
- `modindex`
- `search`



### e

equity\_jenga.api.account\_services.balances,  
4  
equity\_jenga.api.account\_services.inquiries,  
7  
equity\_jenga.api.account\_services.statements,  
5  
equity\_jenga.api.airtime, 8  
equity\_jenga.api.auth, 3  
equity\_jenga.api.exceptions, 20  
equity\_jenga.api.forex, 18  
equity\_jenga.api.kyc, 18  
equity\_jenga.api.loans, 9



## A

`account_details()` (eq-uity\_jenga.api.account\_services.inquiries.AccountInquiry method), 7  
`AccountBalance` (class in eq-uity\_jenga.api.account\_services.balances), 4  
`AccountInquiry` (class in eq-uity\_jenga.api.account\_services.inquiries), 7  
`Airtime` (class in equity\_jenga.api.airtime), 8  
`authenticate()` (eq-uity\_jenga.api.account\_services.balances.AccountBalance method), 4  
`authenticate()` (eq-uity\_jenga.api.account\_services.inquiries.AccountInquiry method), 8  
`authenticate()` (eq-uity\_jenga.api.account\_services.statements.FullStatement method), 5  
`authenticate()` (eq-uity\_jenga.api.account\_services.statements.Ministatement method), 7  
`authenticate()` (equity\_jenga.api.airtime.Airtime method), 8  
`authenticate()` (equity\_jenga.api.forex.Forex method), 18  
`authenticate()` (eq-uity\_jenga.api.kyc.IDSearchVerification method), 18  
`authenticate()` (eq-uity\_jenga.api.loans.CreditScore method), 9  
`available()` (equity\_jenga.api.account\_services.balances.AccountBalance method), 4

## C

`credit_score()` (eq-uity\_jenga.api.loans.CreditScore method),

## 9

`CreditScore` (class in equity\_jenga.api.loans), 9

## E

`equity_jenga.api.account_services.balances` (module), 4  
`equity_jenga.api.account_services.inquiries` (module), 7  
`equity_jenga.api.account_services.statements` (module), 5  
`equity_jenga.api.airtime` (module), 8  
`equity_jenga.api.auth` (module), 3  
`equity_jenga.api.exceptions` (module), 20  
`equity_jenga.api.forex` (module), 18  
`equity_jenga.api.kyc` (module), 18  
`equity_jenga.api.loans` (module), 9

## F

`Forex` (class in equity\_jenga.api.forex), 18  
`forex_rates()` (equity\_jenga.api.forex.Forex method), 18  
`FullStatement` (class in equity\_jenga.api.account\_services.statements), 5

## G

`generate_key_pair()` (in module eq-uity\_jenga.api.auth), 4  
`generate_reference()` (in module eq-uity\_jenga.api.exceptions), 27  
`get_opening_and_closing()` (eq-uity\_jenga.api.account\_services.balances.AccountBalance method), 4

## H

`handle_response()` (in module eq-uity\_jenga.api.exceptions), 27

## I

`IDSearchVerification` (class in `equity_jenga.api.kyc`), 18

## J

`JengaAuth` (class in `equity_jenga.api.auth`), 2, 3

## M

`MiniStatement` (class in `equity_jenga.api.account_services.statements`), 6

## P

`purchase_airtime()` (`equity_jenga.api.airtime.Airtime` method), 8

## R

`retrieve()` (`equity_jenga.api.account_services.statements.FullStatement` method), 5

`retrieve()` (`equity_jenga.api.account_services.statements.MiniStatement` method), 7

## S

`signature()` (`equity_jenga.api.auth.JengaAuth` method), 2, 4

## V

`verify()` (`equity_jenga.api.kyc.IDSearchVerification` method), 19