

# GHFU API DOCUMENTATION

This document serves as an informal documentation to the GHFU API between Martin's front-end and JERM Technology's Backend.

## Server Address:

- IP: xxxx:xxxx:xxxx:xxxx
- Port: YYYYYY
- Protocol: http (https in deployment)

## Data format

All data communication within the API will be via JSON. The only exception is when the request is made from an unknown IP, in which case a simple message *"You are not authorised to access this server!"* and error code 401 will be returned

## API-URLS

### /test

Request JSON: None

Reply JSON: {"status": STRING}

- The STRING should be "Server is up!"

## **/transaction\_status**

Request JSON: {"code":STRING}

Reply JSON: {

    "status":BOOL,

    "log":STRING,

    [opt] "new-code":STRING,

    [opt] "id":INT

}

**Explanation:** The last two keys (*new-code* and *id*) are optional; *new-code* is present if the transaction was successful and so "*status*" is **true**. *id* is only present if the query that initiated the transaction in question was */register*.

### **When is this url called?**

Whenever a call is made to any url involving a fund-transfer (*/buy\_package*, */invest*, */register*, */transfer\_funds*), there are two types of replies that may be returned;

- 1) {"status": true, "code": STRING, "log": ""}: This is returned when the query had no errors. In this case, the system initiates a funds-transfer from the client's mobile phone or from the GHFU Jpesa account to the client's Jpesa account (only for */transfer\_funds*). When the system initiates the fund-transfer, it creates a unique code to be used to get the progress of the transaction. This code is what will be passed to */transaction\_status* to inquire about the progress of the transaction (finished, failed, pending, ...). Typically, one should check the progress of a transaction every 5 seconds.
- 2) {"status": false, "log":STRING}: This is returned when there was an error in the query thus no transaction was initiated by the system. In this case (as always when "*status*" is **false**), refer to "*log*".

### **Why do we need <new-code>?**

In cases like when registering a new member, its very likely that you will need a code to give the new member to feed into the accounts system in order to activate their account. In fact, the new code can even act as the new member's preliminary password. Do whatever you please with *new-code*

**NB:** If the reply is not {"status": false, "log": "pending"}, the transaction code is flushed from the system as the transaction failed for some reason (eg client didn't to pay) or the transaction was successful. An attempt to query about the progress of this transaction will result in a reply of {"status": false, "log": "your code is not in the system"}

## /register

Request JSON:

- ```
{
    "uplink": INT,
    "names": STRING,
    "deposit": FLOAT,
    "number":STRING,
}
```
- eg {"uplink": 3, "names": "Glawn Bukman", "deposit": 1350, "number":"0701173049"}

Reply JSON: see */transaction\_status* about this

- If "id" is 0, error occurred, refer to "log" about the error details. Otherwise, the new account created has ID "id" and will from this point on be addressed by this returned ID

**NB:** In request JSON, "uplink" refers to the new account uplink's account ID and "number" is the phone number (MTN or Airtel) from which the funds are going to be deducted.

## /details

Request JSON: {"id": INT}

Reply JSON: depends;

If reply contains the key "log", error occurred, refer to the log otherwise, the reply will be in the format:

```
{
  "names": STRING,
  "id": INT,
  "uplink": STRING,
  "pv": FLOAT,
  "total_returns": FLOAT,
  "available_balance": FLOAT,
  "total_redeems": FLOAT,
  "commissions": [
    [
      FLOAT amount,
      STRING reason
    ],
    . . .
  ],
  "leg_volumes": [
    FLOAT leg-1-volume,
    FLOAT leg-2-volume,
    FLOAT leg-3-volume
  ],
  "investments": [
    [
      STRING date,
      FLOAT points,
      STRING package,
```

```
    INT package id,  
    INT months_returned,  
    FLOAT returns  
  ],  
  ...  
],  
"direct_children": [STRING child1_names, STRING child2_names, .....]  
}
```

## **/buy\_package**

Request JSON:

```
{  
    "id": INT,  
    "amount": FLOAT,  
    "number": STRING,  
}
```

Reply JSON: see */transaction\_status* about this

### **NB:**

If "id" is 0, then a person not in the system is buying the service/package. Otherwise, a member is buying the package and thus will receive a discount in form of an IBC. Note that if "id" != 0 and the ID does not exist, the system will treat this as though the buyer was not in the system(which is the case if you think about it!).

"number" is the phone number (MTN or Airtel) from which the funds are going to be deducted

## /data\_constants

Request JSON: None

Reply JSON: depends;

If empty (`{}`), then error occurred. Otherwise, reply will be in format:

```
{
  "point-factor": FLOAT,
  "payment-day": INT,
  "account-creation-fee": FLOAT,
  "annual-subscription-fee": FLOAT,
  "operations-fee": FLOAT,
  "minimum-investment": FLOAT,
  "maximum-investment": FLOAT,
  "last-investment-day": INT,
  "exchange-rate": INT,
  "withdraw-charge": FLOAT,
  "rate-inflate": INT
}
```

**last-investment-day:** The day in a month beyond which GHFU does not take new investments from clients. Typically, this value is set to 5

**withdraw-charge:** the % GHFU charges clients when they redeem points from the system

**exchange-rate:** the dollar equivalent used for converting the USD to UGX in the system. An example is 3650.

**rate-inflate:** A value used to twist the *exchange-rate* depending on whether the client is paying GHFU or redeeming funds from GHFU. When GHFU is getting a payment from the client, the actual exchange-rate used is *exchange-rate + rate-inflate*. However, if GHFU is paying a client (client is redeeming funds), the actual exchange-rate used is *exchange-rate - rate-inflate*. This value is typically 50 shillings

## **/set\_data\_constants**

Request JSON:

```
{  
  "point-factor": FLOAT,  
  "payment-day": FLOAT,  
  ...  
}
```

Reply JSON:

```
{  
  "point-factor": BOOL,  
  "payment-day": BOOL,  
  ...  
}
```

### **NB:**

Any number of constants may be omitted in the request JSON. That way, you can even set one constant. Also for any constant NOT in the known list of constants, its corresponding value will be “false” in the reply JSON.

For a full list of constants, see */data\_constants*



## **/invest**

Request JSON:

```
{  
  "id": INT,  
  "amount": FLOAT,  
  "number": STRING,  
}
```

Reply JSON: see */transaction\_status* about this

**NB:** "number" is the phone number (MTN or Airtel) from which the funds are going to be deducted

## /auto-refills

Request JSON:

```
{
  "data":
  [
    [FLOAT points, FLOAT %],
    [FLOAT points, FLOAT %],
    . . .
  ]
}
```

eg

```
{
  "data":
  [
    [375, 59],
    [250, 49],
    [125, 68]
  ]
}
```

Reply JSON: {"status": BOOL, "log": STRING}

- If "status" is "false", error occurred, refer to "log"

**NB:** This URL should be called atleast 2 days before the payment day as the auto-refill percentages for that month are set here!

## **/transfer\_funds**

Request JSON:

```
{  
    "id": INT,  
    "amount": FLOAT,  
    "email": STRING,  
}
```

Reply JSON: see */transaction\_status* about this

**NB:** "email" is the JPesa user-name(email) of the client trying to redeem points. This means that in order for a client to redeem points, they must have a JPesa account. We'd prefer for the client to redeem their funds straight to their mobile money from GHFU but as of now, the JPesa API does not grant us that feature

*All rights reserved by JERM Technology*