Upthinity Technical documentation

Contents

1. Delegate daily and monthly distribution :

Job Location - [root] / controllers / distribute.js

Purpose – The job runs every day on - ‘ 0 0 1 * * * ‘ , which fires `runDailyScripts` and `distributeMonthlyPayments.`

Monthly pay works if its the last day for a month as per the AWS instance timings.
The distribution percentages on how the distribution shall follow -

```
MEMBER_FEE = {
"100-90": 35,
"80-89": 20,
"70-79": 15,
"60-69": 10,
"50-59": 8,
"40-49": 5,
"30-39": 4,
"1-29": 3
}
```

REF: Stat - 1

The Member fee calculation includes “ RANK ” fee structure for members to be distributed Each month. The net Rank for each member is calculated as the ratio for stake time and stake amount or stake amount to stake time whichever is greater for a period of a month.

The logic to divide the award amount among the members sharing the same rank.

The rank thus obtained can be a percentage in REF: Stat – 1 from the total stake offered by a delegate to community.

Note – The calculation is based on “ % “ bases .

```
DEL_FEE = {
"100-90": 40,
"80-89": 30,
"70-79": 15,
"60-69": 10,
"50-59": 5
}
```

REF: Stat - 1

Simillarly, the delegate ranks are calculate as - `(CV + NU + NV) / 3)`

Where CV are the community votes

NU is the node Uptime (Node uptime to be calculated as per the # transactions done per day or time slice particepated in production release)

And NV are the node validations (to be calculated in production release)

The NU and NV are constants for test network with near 100 % values.

Note – The calculation is based on " % " bases .

Note – The job calculates the stats everyday but the distribution takes place at the end of every month.

* The distribution account aka " fee " account credentaials are located at [root] / constants.js

* The fee account for test network collects the fee every hour via a cron job Transaction bot runnnig at [root] / controllers / Txbot.js

* Additional fee is charged via API ' sendPayment ' located at [root] / controller / user.js

*Following are percentages for distribution -

```
 feeUPZ = (1.5 / 100 * (amt))
```

```
Account  Admin
 adminfee = (58 / 100 * (feeUPZ))
```

```
Account Fee
 feefee = (40 / 100 * (feeUPZ))
```

```
Account Burned
 burnedfee = (2 / 100 * (feeUPZ))
```

Location for account credentials - [root] / constants.js

2. Background JOBS

Job Location - [root] / helper / jobs.js

Purpose – The job runs every day on - '0 45 2 * * *' , which fires distributeB
for distribution of stake option 'B'

When a account is staked against a user, everyday on above mentioned time, the expiry is checked and the Job
releases the raw transaction signed by user and escrow.

2.0.1 Stake Configurations

Initial Configurations

Signer - weight: 1 (User)
Weights :
masterWeight: 1, (Escrow)
lowThreshold: 2,
medThreshold: 2,
highThreshold: 2,

Assign User weight and escrow weight to 1:1. i.e, we need both User and Escrow keys to execute any transaction
from now on.
NOTE: Upthinity on test network hold rights for escrow accounts.

Stage 2 Configurations

After initial configurations are submitted, make transactions to :

a) Release Paymnent
b) Revert back config settings to -

masterWeight: 1,
lowThreshold: 0,
medThreshold: 0,
highThreshold: 0,

NOTE: This also makes the account reusable for next stake

2.1 Tx Bot

Job Location - [root] / controllers / tx.js Bot

Purpose – The job runs after every - '0 */59 * * * *' , which calls runPayments
after every 59 minutes to transfer balances from 'txbot' SQL table, loaded with pre funded accounts.
The 'from' and 'to' parameters swap over after each run.

2.2 Money Swap refund

Job Location - [root] / listners / refund.js Bot

Purpose – The job runs after every - '0 */7 * * * *' , calls testRefund.
The purpose of the method is to refund any unmatched transaction either for UPZ - XLM or XLM to UPZ.

NOTE : Please refer system requirements and configurations for more details

3. API’s

Platform offers following API’s (as per repository):

Job Location - [root] / controllers / create.js

3.1 **createAccount**

```
Description : Creates member account along with escrow account
Needs: -
Returns : New Escrow details, New Account details, Recovery passphrase
               & transaction details
```

3.2 **sendRaw**

```
Description : Sends a Raw transaction ( to relese stakes after expiery)
Needs: Raw Tx String
Returns : Transaction details
```

3.3 **createEscrowForStakingB**

```
Description : Creates Stakes for Option B agains the escrow account of user
Needs: Public address, amount
Returns : Transaction details
```

3.3 **createEscrowForStakingA** **(NOT IN CURRENT SCOPE)**

```
Description : Creates Stakes for Option A agains the escrow account of user
Needs: Public address, amount
Returns : Transaction details
```

Job Location - [root] / controllers / delegates.js

3.01 **createAccountDelegate**

```
Description : Creates Delegate account of user
Needs: User Details, Stake Distribution Amount
Returns : Transaction details
```

3.02 **voteDelegate**

Description : Member can use this API to vote Delegates (1 vote per member)
Needs: Delegate Address
Returns : Vote Status

Job Location - [root] / controllers / txBot.js

3.001 **createAccountsRandom** **(SCOPE : Development)**

Description : Creates 50 Random Account with 200 XLM and 999999999 UPZ Starting amount
Needs: -
Returns : Transaction details

3.002 **createAccountsChannles** **(SCOPE : Development)**

Description : Creates 50 Channles Account For sending Transactions via Channles
Needs: -
Returns : Transaction details

3.003 **transferPayments** **(SCOPE : Development)**

Description : Creates Random transactions among 25 accounts
Needs: -
Returns : Transaction details

Job Location - [root] / controllers / user.js

3.0001 **sendPayment**

Description : Sends Payments, for the fee structure refer section **1**
Needs: Sender secret, Reciever public key and Amount
Returns : Transaction details

3.0002 **getDelData**

Description : Gets detail for a delegate account
Needs: Public key
Returns : Account Details

3.0003 **verifyKey**

Description : Recovers user secret
Needs: Public key , Secret passphrase
Returns : Account Secret

3.0004 **trustAsset**

Description : Estlabish trust line for Assets
Needs: Asset Code , Asset Public key , user Secret
Returns : Transaction details