Backend - Api Troubleshooting

What is this document?

In this document are specific backend problems that might happen and the troubleshooting steps in order to fix them.

General considerations

- Backend App's goal is to minimize the amount of errors that require manual intervention. Some fixes presented in this document might be deprecated.
- Always try to look for information in the logs first.
- Every request has a unique id. It's supposed to be used as a filter in logs. Even workers that run enqueued, will keep track of the request_id that originally triggered them.
- Manually triggering funds movements is safe because funds are taken from specific input addresses linked to singular operations; for example, paying one order distributor's commissions can only be funded by that same order payment, even if the Distributor Wallet has funds from other invoices, those funds won't be involved in this Transaction. This maintains accountability very simple as every payment can be linked with each Transaction and address.

Funds are stuck in an address

Scenario 1: Bitgo wallet funds received notification wasn't called

Cause

We have found that sometimes Bitgo doesn't call us when a wallet address has received funds. We are not sure why this happens but we have already reported this to them. As the endpoint and the Tx are public, it's very simple to make the request ourselves.

How to fix

1. Search in the logs if the notification was received. To do so use *Hook received ${TX_HASH}* expression. The transaction hash can be searched using any blockchain explorer[1]. For example: Hook received dbbe58dde6e583a13c1dc78274af2ebed1c81d59c07fd78669a2ccf21bbb415c
2. Based on the search result
    a. If there are no results check if you are searching in a date range containing the transaction date and go back to (1). If it's ok, the notification wasn't received and continue to (3)
    b. If there is only one result, it would probably be the notification without any confirmations. In order to be sure, search for the req_id and you should see *Transaction is unconfirmed.* In this case the notification wasn't received and continue to (3)
    c. If you see one or more notifications and there is at least one that doesn't have *Transaction is unconfirmed* message, this is probably not the error stated in this scenario
3. Using an http client[2] you need to create a post to `APP_ENDPOINT/api/v1/bitgoCallback` (don't forget to set the header `Content-Type: application/json`) with the following body:

```
{
        "hash": "${TX_ID}",
        "type": "transaction",
        "walletId": "${WALLET_ID}"
}
```

[1] http://blockchain.info/, https://live.blockcypher.com/

[2] https://www.getpostman.com/

For example,

```
{
  "hash": "*****************************************************************",
  "type": "transaction",
  "walletId": "***********************************"
}
```

4. Once done you will receive *STATUS 200* and funds will be moved. You will be able to see the corresponding transaction in a blockchain explorer.

Scenario 2: Sales app rejected funds movement

Cause

If the amount received does not correspond to the one expected by the sales app (based on ticket amount) the transaction won't be placed.

How to fix

1. Search in the logs if the error actually happened. To do so use *Hook received ${TX_HASH}* expression. The transaction hash can be search using any blockchain explorer[3]. For example: *Hook received* **
2. Searching using the req_id found in (1) you should be able to see *"Amount received is invalid for ticket"*
3. Update Sales App db in order to allow fund movement
4. Trigger the notification (as described in Scenario 1 - Step 3). *Note: This will be done automatically once sales app allows this error to be retried.*

[3] http://blockchain.info/, https://live.blockcypher.com/

Scenario 3: Funds cannot be moved because a whitelisting denial

Cause

When Trying to move funds bitgo rejects the transaction because one of the destination addresses is not in the whitelist. This seems to be a bitgo error when calling add to whitelist simultaneously.

How to fix

1. Search in the logs if the error actually happened. To do so use *Hook received ${TX_HASH}* expression. The transaction hash can be search using any blockchain explorer[4]. For example: *Hook received* **
2. You should be able to see *Error: denied by policy*
3. Perform GET `APP_ENDPOINT/api/v1/invoiceWallets/${WALLET_ID}` and check if all the outputs are contained in admin.policy.rules[0].condition.addresses
4. Using IOHK custom Bitgo-cli[5] whitelist the missing addresses
    a. Login: `bin/bitgo login`
    b. Select the wallet to add whitelist to: `bin/bitgo addWhitelist`
    c. Input the address to whitelist `***********************************`
    d. Add whitepolicy: `bin/bitgo ***********************************`
    e. You sould see a `'Done!'` message
5. Backend App will retry and send the funds

[4] http://blockchain.info/, https://live.blockcypher.com/

[5] https://github.com/atixlabs/bitgo-cli

Bitgo API token max spend exhausted

Cause

For each API Token Bitgo allows a certain amount of total BTC to be sent.

How to fix

1. Search in the logs if the error actually happened. To do so use *Hook received ${TX_HASH}* expression. The transaction hash can be search using any blockchain explorer[6]. For example: *Hook received* **
2. You should be able to see *Error: needs unlock*
3. There are 2 possibilities
   a. Contact Bitgo and ask them to increase the limit
   b. Issue a new token
      i. Decrypt production.json file
      ii. Change the Bitgo token with the new one
      iii. Encrypt production.json file
      iv. Push changes
      v. Release and deploy a new version

[6] http://blockchain.info/, https://live.blockcypher.com/

Exodus address needs to be changed

Considerations

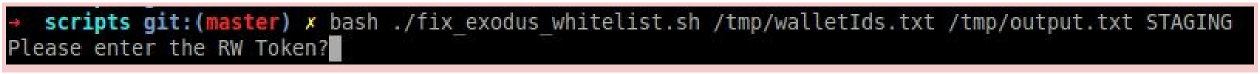
When the exodus address is changed, all existing white-policies from invoices Wallets need to be updated as well.

Depending on the amount, we could ask Bitgo people to make a massive update, or run a script ourselves. Last time (Oct 2016), we had ~6000 wallets and they ask us to run the script.

Steps to Proceed

1. Make sure there are no current transactions and sales are stopped
2. Update *production.js* settings with the new exodus in **master** branch
3. Deploy the new version to production (Build with *Jenkins* and deploy with *Rundeck*)
4. Export all Production Wallets Ids to a text file. **(*)**
    a. Connect to **Backend** Production DB
    b. execute: `` `SELECT DISTINCT walletid FROM keys` ``
    c. Export the result to a text file (each line should be a wallet Id, nothing else).
5. Run Script fix_exodus_whitelist.sh (the script is in *./scripts/* folder, not *./src/scripts/*)
    a. Param $1: File Path to Wallet Ids list
    b. Param $2: File Path to output activity log (will be created if not exist)
    c. Param $3: Environment to run, Options = {‘STAGING’, ‘PRODUCTION’}
    d. The script will ask for the **Production RW TOKEN**

    Example: `` `bash ./fix_exodus_whitelist.sh /tmp/walletsIds.txt /tmp/output.txt STAGING` ``



```
→  scripts git:(master) ✗ bash ./fix_exodus_whitelist.sh /tmp/walletIds.txt /tmp/output.txt STAGING
Please enter the RW Token?
```

6. **Validate Whitelistings**: the script hits a Backend endpoint that enqueues all whitelisting requests, so the script finishing is not an indicator that we are ready to go. You should also check the whilisting queue evolution. That can be checked in the DB, whitelisting table.

(*) Actually this can also be found in Sales-App DB, and probably these wallets can be sorted in a way that the first ones are the Most Recently Used. If run in that order, they would be processed first and can be ready for use without waiting the whole script and the whitelisting process to finish. This could be particularly beneficial if the exodus address switching needs to be done on a live sales scenario, where MRU Wallets are more prone to receive new orders first.