Transactions

This document provides a short intro into transactions between two addresses. For a simple explainer on blockchain transactions, go here

Prerequisites

Tools required for the process are:

- cardano-cli
- source address with some funds (see faucet, or request funds)
- network

If you started local tools from provided doker compose file, cardano-cli is available in almost any container, but maybe easiest to use is `apex-relay`.

Some notes:

- when inside docker-container both `/genesis` and `/config` folders are visible both on host and in the container (only need sudo to manipulate them while in host). This is a shortcut to manipulate keys and transactions without the need to send back and forth the data between container and host.

- All commands marked with (Online) need to be executed on the container (or running node socket available) and (Offline) can be executed anywhere including on the container but does not require active node to function properly.

- To get the easy playground, feel free to put key files in the **`/config`** folder, say subfolder **`test`**, then exec into the container change to **`/config/test`** directory and execute all command from there. This is least secure way as secret signing key is exposed, but reduces the overhead to sign transactions in an offline environment and send them over, in test env shold be all good.

- You can execute container's **`cardano-cli`** from host as well, for example to check the tip you can use:

```
docker exec -it vector2-testnet-tools-v3-apex-relay-1 cardano-cli que
  --testnet-magic 1177 \
  --socket-path /ipc/node.socket
```

- The **`cardano-cli`** command outputs the table format, but it can also output json as well with **`--out-file /dev/stdout`** suffix as it outputs the json format when requested to save to file. This is handy for data extraction, see json section at the bottom.

All commands below are given relative to **`cardano-cli`** executable regardless of calling it from host ot container, and will produce results in current folder.

```
NOTE: --testnet magic should be set to the appropriate number. Please
$ export TESTNET_MAGIC="<CORRECT VALUE HERE>"
```

Blockchain info

Query tip (Online):

```
cardano-cli query tip \
  --testnet-magic $TESTNET_MAGIC \
  --socket-path /ipc/node.socket
```

Keys handling

Generate a new payment key pair (Offline). Those are used for funds send and receive. Result will be keys in following formats:

- one Payment Verification Key: `PaymentVerificationKeyShelley_ed25519`
- one Payment Signing Key: `PaymentSigningKeyShelley_ed25519`

```
cardano-cli address key-gen \
  --verification-key-file test0.payment.vkey \
  --signing-key-file test0.payment.skey
```

Generate a new stake key pair (Offline). Those are used for protocol participations and rewards. Result will be keys in following formats:

- one Stake Verification Key: `StakeVerificationKeyShelley_ed25519`
- one Stake Signing Key: `StakeSigningKeyShelley_ed25519`

```
cardano-cli stake-address key-gen \
 --verification-key-file test0.stake.vkey \
 --signing-key-file test0.stake.skey
```

Address handling

Generate the payment address based on the payment verification key. (Offline) Result will be a file with a single address.

```
cardano-cli address build \
  --testnet-magic $TESTNET_MAGIC \
  --payment-verification-key-file test0.payment.vkey \
  --out-file test0.payment.addr
```

Generate the stake address based on the stake verification key. (Offline) Result will be a file with a single address.

```
cardano-cli stake-address build \
  --testnet-magic $TESTNET_MAGIC \
   --stake-verification-key-file test0.stake.vkey \
  --out-file test0.stake.addr
```

Check the balance of generated address to make sure it is 0 (Online):

```
cardano-cli query utxo \
  --address $(cat test0.payment.addr) \
  --testnet-magic $TESTNET_MAGIC \
  --socket-path /ipc/node.socket
```

Transaction handling

The main goal is to get the fee and invalid-hereafter. For the fee we will take 1 apex (1000000 dfm) and for invalid-hereafter we will check the **`slot`** from the tip and add some value depending on the expectd validity of the transaction.

Check the transaction hash for source address (Online):

Note that we need to have both source and target address... Here is given example with test0.payment.addr which is topped up from faucet and some imaginary target.payment.addr which is generated with the above steps.

For the example in this case currently test0 address value is **`addr_test1vq2qpmqwz8ckdw85krtmsmf3u9j62nzfskvtczn4gtedwrsdms62f`** and it has TxHash of **`d11b9e81deb9806104f0251ba911897fade943fb3f0fe743425e2b978468f3a1`** and index is TxIx of 0. This is ephemeral state and will change when utxo advances...

```
cardano-cli query utxo \
  --address  $(cat test0.payment.addr) \
  --testnet-magic $TESTNET_MAGIC \
  --socket-path /ipc/node.socket
```

Calculate the change (Offline):

Example is from 500 mil we send 20k with fee of 1, change is 499979999

```
expr 500000000000000 - 20000000000  - 1000000
```

Next we will have the **`slot`** for invalid-hereafter. It is some slot in the future, after which transaction is invalid even if properly signed and submitted. Depending on the speed of the network, value can vary, but if it is submitted too slow it will be rejected if submitted after slot listed in invalid-hereafter. To get the current slot we query the tip, and then add some offset, say 5000, or 1000 depending on how fast we will sign and submit the transaction.

Query tip (Online):

In this example current slot value was 152596 and we chose to add 5000 to it to get value of 157596 for invalid-hereafer.

```
cardano-cli query tip \
  --testnet-magic $TESTNET_MAGIC \
  --socket-path /ipc/node.socket
```

Create raw transaction (Offline): Result is **`Unwitnessed Tx BabbageEra`**, note that TxHash has to have index appended (#0) and most often change address is the source address (although we can send change to the arbitrary address). Also note that we are having here two outputs (**`tx-out`**) one for target address and one for change (in format address+amount) but we can have multiple as well ant they will all share the same fee, generating multiple utox outputs. They can be directed to the same, mix or completly different addresses.

```
cardano-cli transaction build-raw \
  --tx-in d11b9e81deb9806104f0251ba911897fade943fb3f0fe743425e2b97846
  --tx-out $(cat target.payment.addr)+20000000000 \
  --tx-out $(cat test0.payment.addr+499979999000000 \
  --fee 1000000 \
  --invalid-hereafter 157596 \
  --out-file tx-1.raw
```

Sign the transaction (Offline): Result is **`Witnessed Tx BabbageEra`**

```
cardano-cli transaction sign \
  --tx-body-file tx-1.raw \
  --signing-key-file test0.payment.skey \
  --testnet-magic $TESTNET_MAGIC \
  --out-file tx-1.signed
```

Submit the transaction (Online):

```
cardano-cli transaction submit \
  --testnet-magic $TESTNET_MAGIC \
  --socket-path /ipc/node.socket \
  --tx-file tx-1.signed
```

Check the source address (Online):

```
cardano-cli query utxo \
  --address $(cat test0.payment.addr) \
  --testnet-magic $TESTNET_MAGIC \
  --socket-path /ipc/node.socket
```

Check the destination address (Online):

```
cardano-cli query utxo \
  --address $(cat target.payment.addr) \
  --testnet-magic $TESTNET_MAGIC \
  --socket-path /ipc/node.socket
```

Minimum fee

You can also check minimum fee (we used fee of 1 apex above) instead.

get protocol parameters

```
cardano-cli query protocol-parameters \
  --testnet-magic $TESTNET_MAGIC \
  --out-file misc/protocol.json \
  --socket-path node-ipc-1/node.socket
```

calculate min fee for the given transaction parameters and raw transaction

```
cardano-cli transaction calculate-min-fee \
  --tx-body-file transactions/tx-faucet-1.raw \
  --tx-in-count 1 \
  --tx-out-count 2 \
  --witness-count 1 \
  --byron-witness-count 0 \
  --testnet-magic 1177 \
  --protocol-params-file misc/protocol.json
```

JSON handling in `cardano-cli`

Some shortcuts to get data out of **`cardano-cli`** responses using the **`jq`** tool. Note tha some of the commands of **`cardano-cli`** do not produce json output. Fot them to work with json **`--out-file`** should be used and can be piped further if directed to **`/dev/stdout`**.

Percentage of node synced (should be **`100.00`** for fully sunced node):

```
cardano-cli query tip  \
  --testnet-magic $NETWORK_MAGIC \
  --socket-path $NODE_SOCKET | jq -r .syncProgress
```

Get total balnce of all UTXOs on the address:

```
cardano-cli query utxo \
  --address  $ADDRESS \
  --testnet-magic $NETWORK_MAGIC \
  --socket-path $NODE_SOCKET \
  --out-file /dev/stdout | jq -r '. | map(.value.lovelace) | add'
```

Get the hereafter and add some delta to it:

```
cardano-cli query tip \
  --testnet-magic $NETWORK_MAGIC \
  --socket-path $NODE_SOCKET | expr $(jq .slot ) + $HEREAFTER_DELTA
```
