Using SmartBCH via Mainnet Cash

A Quick Start

2qx - github.com/2qx

2022-06-20 / HackSmart Dojo

Outline

Suggested Tooling

To Follow Along On Your Own Battle Station

- These slides: https://bit.ly/3MZEXJq
- A computer with a POSIX command prompt and some GNU tooling, i.e. (Linux/WSL/Darwin)
- Docker & docker-compose
- The `mainnet-js` source code
  `git clone https://github.com/mainnet-cash/mainnet-js.git; cd mainnet-js`
- Run `yarn regtest:sbch:up` from the mainnet-js root directory to pre-download & deploy a RegTest network stack.
- Some kind of nodejs, or python http serverlet
  `python3` or `npm i reload -g`
- Browser, with Metamask installed

Mainnet Cash is a Specification

First and foremost.

Mainnet Cash is a tool for developers to access Bitcoin Cash

- A contract for how a service will be provided.
- Generated and enforced to the specification programmatically, to the extent practical
- The specification: api.yml
- Specifically, an OpenAPI 3 spec, implement in typescript, used in a generated express server.

Developer convenience and expediency are valued over perfection, code cleanness or symmetry.

Starting your own mainnet.cash

To start your own mainnet.cash REST service

```
# get the image from docker
docker pull mainnet/mainnet-rest

# start the image, with one process,
# exposing locally at 3000 to the container's port 80
docker run -d --name Mainnet --env WORKERS=1 \
           -p 127.0.0.1:3000:80 \
           mainnet/mainnet-rest
```

Stopping your own mainnet.cash

Just in case something **goes wrong**...

```
# checking logs, to find the issue
docker logs -f Mainnet

# restarting, stopping, forcefully stopping.
docker restart Mainnet
docker stop Mainnet
docker kill Mainnet

# if your machine is paging memory or sounds like a jet plane
sudo killall node
# kills all docker processes
docker stop $(docker ps -a -q)
```

assuming your container was given the name "Mainnet".

More Usage Samples and Configuration Information

- There's more at: mainnet.cash/tutorial/running-rest.html
- Including a full list of configuration variables.
- This is an unstable version of the REST service for testing and educational purposes

Running a RegTest Stack

The mainnet-js source code contains a full RegTest Stack, deployed with docker compose.

```
# get the mainnet-js source
git clone https://github.com/mainnet-cash/mainnet-js.git

# enter the root package
cd mainnet-js

#  To start the stack, call from the mainnet-js repo
yarn regtest:sbch:up

# reset everything
yarn regtest:sbch:down; yarn regtest:sbch:up
```

There's more in the docs.

Having your own private networks allow for better tests, testing in time, testing with a RegTest Bitcoin Cash chain.

Note: the docker images for the stack took about 6 min to download over wifi.

Here we wait... a random historical anecdote

In 1785 a French mathematician named Charles Joseph Mathon de la Cour wrote a parody of Benjamin Franklin's Poor Richard's Almanac in which he mocked the unbearable spirit of American optimism represented by Franklin. The Frenchman fictionalized about "Fortunate Richard" leaving a small sum of money in his will to be used only after it had collected interest for 500 years. Franklin, who was seventy-nine years old at the time, wrote back to de la Cour, thanking him for the great idea and telling him that he had decided to leave a bequest to his native Boston and his adopted Philadelphia of 1,000 pounds to each, on the condition that it be placed in a fund that would gather interest and support the public good for the succeeding 200 years.

The RegTest Secrets

One of the advantages of RegTest is reusing and sharing secrets

```
ALICE_ID="privkey:regtest:0x758c7be51a76a9b6bc6b3e1a90e5ff4cc27aa054b77b
ALICE_ADDRESS="0xE25ddbAF8DD61b627727e03e190E32feddBD1319"
BOB_ID="seed:regtest:total unlock silver actual hunt excuse tone hen fix
BOB_ADDRESS="0xaf9aD25ea3603B0aF520e1B5331d87315B6eB72B"
```

Run the packages from a Browser

When developing from a jsfiddle or codepen...

```
<!-- mainnet-js -->
<script src="https://cdn.mainnet.cash/mainnet-0.5.4.js"
        integrity="sha384-W9XdLRFn3qb7qn1+gqFho+FTw9buULZBaQh+uceAAMmK3w
        crossorigin="anonymous">
</script>
<!-- CashScript -->
<script src="https://cdn.mainnet.cash/contract/contract-0.5.4.js"
        integrity="sha384-4htLbqsKWOCH8CvROESHkSDw11RiDXodGnw7e2xc3mcArv
        crossorigin="anonymous">

</script>
<!-- SmartBCH -->
<script src="https://cdn.mainnet.cash/smartbch/smartbch-0.5.4.js"
        integrity="sha384-sk/jV4NeaEsrfMShEiO3C8zMwEOHzwfrVu6s6EiVoqOmnU
        crossorigin="anonymous">
</script>
```

Run the packages from a Browser

We want to load all the most recent bundles in a browser to access the library from the dev console...

```
# open a terminal and cd to the mainnet-js directory
cd projects/mainnet-js

# serve an empty app with the smartBCH bundle preloaded
python3 -m http.server -d jest/playwright/smartbch/

# python3 may be called python,
```

`npx reload` or similar is fine too. The script `yarn live` is a shortcut

You should now have a browser console with the latest bundles on port 8000.

Simply pull mainnet-js to update versions.

Get a wallet

If you **didn't get RegTest** harness working...
Create a TestNet Wallet and get the deposit address:

```
const alice = await TestNetSmartBchWallet.newRandom();
//
await alice.getBalance()
// Object { bch: 0, sat: 0, usd: 0 }
alice.getDepositAddress()
// "0xE25ddbAF8DD61b627727e03e190E32feddBD1319"
alice.toString()
// "seed:t........"
```

Save the wallet string, use it again with `TestNetSmartBchWallet.fromId()`

Getting Testnet sBCH

If you are **not** using RegTest, you can obtain about 0.1 sBCH here:

- There is a TestNet faucet here
- 54.169.31.93:8080/faucet

On site users should skip this step if they have RegTest configured.

Get a wallet

Create a RegTest Wallet and get the deposit address:

```
ALICE_ID="privkey:regtest:0x758c7be51a76a9b6bc6b3e1a90e5ff4cc27aa054b77b
const alice = await RegTestSmartBchWallet.fromId(ALICE_ID);
//
await alice.getBalance()
// Object { bch: 10000, sat: 1000000000000, usd: 1141000 }
alice.getDepositAddress()
// "0xE25ddbAF8DD61b627727e03e190E32feddBD1319"
alice.toString()
// ALICE_ID
```

Sending/Watching

Watch Bob's address as Alice sends coins:

```
ALICE_ID="privkey:regtest:0x758c7be51a76a9b6bc6b3e1a90e5ff4cc27aa054b77b
alice = await RegTestSmartBchWallet.fromId(ALICE_ID);
const bob = await RegTestSmartBchWallet.newRandom();

const bobBalanceWatchCancel = bob.watchBalance((balance) => {
    bobBalanceWatchCancel();
    console.log("bob got paid!")
});

alice.send(
    { address: bob.getDepositAddress(), value: 0.001, unit: "bch" },
    {},
    { gasPrice: 10 ** 10 }
);
```

Call Metamask

This command will cause your browser to connect to Metamask, on TestNet

```
let alice = await Web3TestNetSmartBchWallet.init();
```

- Official Documentation from SmartBCH
- Network URL: https://moeing.tech:9545
- The chain ID is: 0x2711 (i.e. 1001)

Call Regtest Metamask

Repeat the step above, but for your private RegTest network.

```
let alice = await Web3TestNetSmartBchWallet.init();
```

- Network URL: http://127.0.0.1:8545
- The chain ID is: 0x2712 (i.e. 1002)

Committed to Adoption

Mainnet Cash will be here.

- The project **has some remaining funds to handle maintenance**.
- Currently in a **feature freeze**.
- Seeking **broader adoption**, and usage.
- The project is suitable for small or limited value transactions. $0.02-$50
- An independent **security audit** would be in order before the warnings are removed.

Warnings aside, mainnet-js was used for the old hop.cash and is in production on old.hop.cash.

The Future

Mainnet Cash was built to be maintained.

- Really conservative and maintenance oriented design choices from the start.
- Extensive investment in testing and a testing harness
- These values are also apparent in supporting libraries.
- Outlook
  - **@bitauth/libauth** has a nice upgrade in the works.
  - There are exiting things are happening with both SmartBCH and BCH

Contact

- More documentation at mainnet.cash
- Please report Bugs on Github.
- Join our mainnet.cash Telegram Channel for support.