Script TV - nodeops - stvtool user manual

stvtool v1.10

Introduction

This document explains the setup and high-level operation of a script.tv node. In order to test the functions it is convenient to install a testnet node, which allows you to obtain free funds from the testnet-faucet. With testnet SCPT tokens wou'd be able to test functions requiring token transfers, e.g. buying licenses.

In Mainnet installations the faucet service is not available and tokens must be acquired from a different route. e.g. Exchanges.

1-liner installer

A command meant to be executed in a fresh debian GNU/Linux-12 operating system.

```
wget -O install.sh https://download.script.tv/files/script_tv-node-testnet_debian_11_x86_64__install.sh; chmod +x install.sh; ./install.sh [<Arguments>]
```

Optional Arguments: ### -batch

- Non interactive mode suitable for inclusion in automated processes.

--node_key <64 byte hex>

- Set the node secret key instead of generating a new random one.

Node setup

Use a debian GNU/Linux 12 (or compatible) fresh installation as starting point. Decide which network node you wish to install and execute the 1liner installer:

Default node setup. Generated secret key.

For testnet:

```
wget -O install.sh https://download.script.tv/files/script_tv-node-testnet_debian_11_x86_64__install.sh; chmod +x install.sh; ./install.sh
```

For mainnet:

```
wget -O install.sh https://download.script.tv/files/script_tv-node-mainnet_debian_11_x86_64__install.sh; chmod +x install.sh; ./install.sh
```

Node setup with assigned secret key

The node key can be pre configured by passing it to the installer using `--node_key <64 bytes Hex format>` Examples:

For testnet:

```
wget -O install.sh https://download.script.tv/files/script_tv-node-testnet_debian_11_x86_64__install.sh && chmod +x install.sh && ./install.sh --node_key 0102030405060708090a0b0c0d0e0f101112131415161718191a1b1c
```

For mainnet:

```
wget -O install.sh https://download.script.tv/files/script_tv-node-mainnet_debian_11_x86_64__install.sh && chmod +x install.sh && ./install.sh --node_key 0102030405060708090a0b0c0d0e0f101112131415161718191a1b1c
```

In both modalities, these commands will download and run the installation package.

This screencast recording illustrates the process: https://x.com/manic_beret/status/1865704546011377953

This program will install the Script TV node software on this computer.

The installer is staright forward and when finish it will leave a running node.

System Overview

After a successful installation the node is operations, all services up and running. There are two linux users

- root: used for bringing up or down services, checking system logs and any activity in the scope of Operating System maintenance.
- stv: used for running script.tv services and user tool stvtool.

system user: root

```
Use ssh to access a super-user shell by ssh the IP address of the node. sshd runs on TCP port 22 (default ssh port).

        ssh root@<ip-address>
```

files and directories

Technical datasheet

```
Contains information about:
        * systemd daemons
        * Open TCP Ports
        * URLs
        * configuration files for the different subsystems

        Access the document typing:

        cat /var/script_tv/data_sheet
```

systemd daemons

```
List all script_tv daemons installed:

        ls -la /etc/systemd/system/script_tv*

inspect any daemon to check launch parameters, e.g. it can be verified all daemons run as unpriviledged user stv:

        cat /etc/systemd/system/script_tv__<subsystem>.service

script_tv__\<subsystem\>.service are systemctl service identifiers.
        verify daemon status:

        systemctl status <service identifier>

start/stop subsystems:

        systemctl stop <service identifier>
                systemctl start <service identifier>

All script_tv subsystems can be stopped and started again all at once for system maintenance purposes:

        script_tv__ctl.sh stop
                script_tv__ctl.sh start

Daemon log files are set to produce only error information (loglevel) reducing execution overhead.
        Read logs for a given service identifier using journalctl:

        journalctl -u <service identifier>
```

L1 subsystem - gov

```
The governance daemon is responsible of building the blockchain and execute the P2P consensus protocol with other nodes.
        The main binary is located at:

        /usr/local/bin/script_tv__script4__gov
```

It is launched by systemd using the service identifier script_tv__be__L1__script4__gov.service located at /etc/systemd/system/ Its working directory is at /home/stv/script4/gov/ Inside this directory we find the daemon configuration file config.yaml. This file contains:

- hash of the genesis block
- chain name, either mainnet or testnet
- Ethereum compatibility chain id, either 62855 or 62854
- license directory, where the daemon checks for validator/lightning licenses.
- P2P consensus protocol (public) and RPC TCP ports (open for 127.0.0.1)
- seeds
- NAT mapping, disabled by default
- Log level. Possible values in verbosity order are Trace, Debug, Info, Warn, Error, Fatal, and Panic

Any change in this file should be followed by a daemon restart:

```
systemctl restart script_tv__be__L1__script4__gov.service
```

The directory /home/stv/script4/gov also contains: * directory db/: Database where the blockchain is built. * directory key/: Where the node secret key is. This key is the seed for the node public key and address. * file snapshot: Current ledger snapshot, initially it matches the genesis block but is update on checkpint and used as baseline.

The shell command *script* is an alias for

```
/usr/local/bin/script_tv__script4__gov --config=/home/stv/script4/gov
```

L1 subsystem - wallet

```
The wallet daemon is responsible for managing user keys and signing transactions.
```
The main binary is located at:

```
/usr/local/bin/script_tv__script4__wallet
```

It is launched by systemd using the service identifier script_tv__be__L1__script4__wallet.service located at /etc/systemd/system/ Its working directory is at /home/stv/script4/wallet/ Inside this directory we find the daemon configuration file config.yaml. This file contains:

- URL for reaching the gov daemon http/RPC service
- chain name, either mainnet or testnet
- Ethereum compatibility chain id, either 62855 or 62854 Any change in this file should be followed by a daemon restart

```
systemctl restart script_tv__be__L1__script4__wallet.service
```

The directory /home/stv/script4/wallet also contains: * directory keys/: Where the user wallet keys are stored.

The shell command *scriptcli* is an alias for

```
/usr/local/bin/script_tv__script4__wallet --config=/home/stv/script4/wallet
```

Uninstall

```
Execute the program
```

```
${system_unix_name}__uninstall.sh
```

```
This will stop running services and delete all installed files from the hard disk.
```

script-tv user: stv. stvtool

This linux account is used to operate the services provided by L1 daemons via their RPC and CLI interfaces.

from root shell change to user stv via a helper program located at /usr/local/bin/srv:

```
stv
```

or, using OS tool su:

```
su - stv
```

Verify the current directory changed to */home/stv* and the current user changed to *stv*

```
pwd
        /home/stv
        whoami
        stv
```

The main command is *stv*, which is an alias for the program `bin/stvtool`, a program written in *bash*.

A simple invocation provides with a comprehensible list of functions

```
stvtool - Script-Network user tool - v1.10
    Network: testnet

    --- Underlying programs:
        * script (L1 governance / blockchain)
          /usr/local/bin/script_tv__script4__gov --config=/home/stv/script4/gov

        * scriptcli (local wallet)
          /usr/local/bin/script_tv__script4__wallet --config=/home/stv/script4/wallet

    Usage:
    stvtool [options] <command>
    options:
         -a ..................................... Advanced user/extended output.
         -batch ................................. Non-interactive mode for automation.
                                                  Commands produce parse-fiendly output.

    <command>:
    --- General
        * docs ................................. Show information related to documentation and logs

    --- Local
        * ip4 .................................. Print LAN and WAN IPv4 addresses.
        * endpoints ............................ List private services and ports, and public URLs
        * test ................................. Run tests.
        * take_snapshot ........................ Create a snapshot block.
        * fetch_snapshot ....................... Obtains a snapshot block from other node.

    --- Network
        * [-a] status .......................... L1 Status, parameters and node public key.
        * peers ................................ Peers connected to the node on the network.

    --- Wallet keys
        * keys ................................. List wallet public keys.
        * keysp ................................ List wallet keys including private keys.
        * new_key .............................. Generate and store a new key pair.
        * import_key <hex-64>................... Generate and store a new key pair from a given private key.
        * backup ............................... Produce a tgz with gov and wallet keys.

    --- Tokens. SCPT and SPAY.
        * balance .............................. Token Balances. Defaults to node address balance.
            [--address <address>] .............. The address to show balance for.
        * transfer ............................. Move tokens
            --from <address> ................... Source address.
            --to <address> ..................... Recipient address.
            --scpt <amount> .................... SCPT amount 0.000000000
            --spay <amount> .................... SPAY amount 0.000000000

    --- Faucet
        * faucet ............................... Command for receiving free SCPT and SPAY. Options:
              (default node address: 0xD9e639663d42A18D5489bbb25317BA509fb968ba (this node))
            --recipient <address> .............. Recipient address
            --lightning ........................ Obtain funds to run a lightning node.
            --validator ........................ Obtain funds to run a validator node.

    --- Node types and licensing
```

```
        * buy_license ................................ Purchase license to run a validator node.
            --for <address> ........................ The address to buy the license for.
            --from <address> ....................... The address to pay license fee from.
            --referral <address> ................... The referral address.
            --lightning ............................ Buy a lightning license.
            --validator ............................ Buy a validator license.
        * redeem ..................................... Redeem easeFlow purchase.
        * print_license .............................. Print the license.
              (Default node address: 0xD9e639663d42A18D5489bbb25317BA509fb968ba (this node))
            --all ..................................  Print all licenses.
            --lightning ............................ Print lightning license.
            --validator ............................ Print validator license.
            --address <address> ..................... ..The address to print license for.
        * fetch_licenses ............................. Download the licensee database.
        * print_referral ............................. Print the referral message to share.

    --- Staking
        * stake ...................................... Command for Staking. Options:
            --lightning ............................ Stake 20K SCPT. Become a Lightning node.
            --validator ............................ Stake 1M SCPT. Become a Validator node.
            --amount ............................... Stake amount other than default amounts.
        * delegate ................................... Stake other nodes.
            --delegate_to .......................... The node address to delegate to.
            --amount ............................... SCPT amount to stake on delegated node.
        * node_status ................................ Node status summary.
            --address <address> ..................... The address to check status for.

        Network: testnet
```

Functions are grouped in the following sections: * Local. Functions related to this node. * Network. Functions related to the network of nodes. * Faucet. Functions related to faucet. Receive free tokens. Only available on testnet. * Wallet. Functions related to tokens. * Licensing. Functions related to licenses. * Staking. Functions related to staking.

The option `-batch` provides output suitable for being parsed by higher level automation.

Local node functions

stv ip4

Outputs the LAN and WAN IPv4 addresses of this node. Possible output:

```
$ stv ip4
        WAN 16.134.4.199
        LAN 192.168.0.123
```

stv endpoints

```
Provides the following information

$ stv endpoints
        PUBLIC (LAN/WAN interface)
        --- Open TCP ports:
            * script4 governance (ledger, blockchain, consensus). ............ <Public TCP Port>

        --- URLs:
            * script4 governance json RPC backend ............................ <Public RPC endpoint.https://>
            * wallet json RPC backend ........................................ <Public RPC endpoint.https://>

        PRIVATE (localhost interface)

        --- Open TCP ports:
            * script4 local wallet ........................................... <Private TCP Port>
            * script4 governance json RPC backend ............................ <Private TCP Port>

        --- Endpoint URLs:
            * script4 governance json RPC backend ............................ <Private RPC endpoint.http://>
            * script4 local wallet json RPC backend .......................... <Private RPC endpoint.http://>
```

stv test

Executes a series of health checks reporting any malfunction (KO) or otherwise (ok). Example output:

```
$ stv test
        ok be/b2c daemon nginx: Active: active (running)
        ok be/explorer daemon nginx: Active: active (running)
        ok be/L1/bridge_eth daemon nginx: Active: active (running)
        ok be/L1/script4 daemon nginx: Active: active (running)
        ok be/wallet daemon nginx: Active: active (running)
        ok db/explorer daemon mongod: Active: active (running)
        ok fe/explorer daemon nginx: Active: active (running)
        ok fe/wallet daemon nginx: Active: active (running)
        ok be/L1/script4 ports_local port 10000/tcp open. be/L1/script4 1 10000 consensus all
        ok be/L1/bridge_eth daemon script_tv__be__L1__bridge_eth: Active: active (running)
        ok be/wallet/mainnet/genesis ports_local port 443/tcp open. be/wallet/mainnet/genesis 2 443 https all
        ...
```

stv take_snapshot

Creates a snapshot block at the current height. The location where it is placed is printed on screen.

Possible output:

```
stv take_snapshot
        /home/stv/script4/gov/backup/snapshot/script_snapshot-216040-0x0b1d763ae9718fb01e43d7d1d421839e3b2ea020ce1879809e88ffedccf4a40f-2024-12-08
```

stv fetch_snapshot

Fetch a snapshot block from other node. The location where it is placed is printed on screen. Saves the currently in use snapshot found at /home/stv/script4/gov/snapshot as gov/backup/attic/snapshot_-timestamp-, before replacing it with the obtained one. If the transmission fails the original snapshot file is returned back to its original place.

Possible output:

```
stv fetch_snapshot
        Saved current snapshot as /home/stv/script4/gov/backup/attic/snapshot_20241208215410
        Obtained a snapshot at /home/stv/script4/gov/snapshot
        Restart the gov service: (as root) systemctl restart script_tv__be__L1__script4__gov
```

Network functions

stv status

Provides networking information related to this node

- chain_signature: unique string identifying the blockchain instance (tied to unique genesis block)
- node_address: This node address, derived from the key found in directory /home/stv/script4/gov/key
- peer_id: Alternative node identification for the context of neighbour nodes.
- current_height
- explorer_progress_height

stv -a status

Provides extended networking information related to this node, and blockchain status

- chain_id: either mainnet/testnet
- current_epoch: ordinal for the current block finalization epoch
- current_hash: current block hash
- current_height: current block height

- current_time: current unix time
- eth_chain_id: Numeric identifier for ethereum compatibility
- genesis_block_hash: hash of the genesis block
- latest_finalized_block_epoch: last epoch already finalized
- latest_finalized_block_hash: latest finalized block hash
- latest_finalized_block_height: latest finalized block height
- latest_finalized_block_time: latest finalized block unix time
- snapshot_block_hash: ledger snapshot hash
- snapshot_block_height=height corresponding to the snapshot
- syncing: false if the node is currently working on the latest block True if it is catching up.
- tip_hash: latest syncchronized block.
- explorer_progress_height: latest block available in the blockchain explorer.

Possible output in batch mode:

```
$ stv -batch -a status
        address                         0x8a95c6C85A4eB294c4dCB764867717C7CDf1f072
        chain_id                        testnet
        current_epoch                   9144
        current_height                  9142
        current_hash                    0x5e794a6d03b3e2012cb2b7912e137407029c7dcf2a68fb8f08628c9c3c20edbe
        current_time                    1734180003
        eth_chain_id                    42854
        genesis_block_hash              0x3a3fc4ef5d3e7509006963a30ee3d04f60d24645d56b7ea7d8403b9b6ff8d62a
        latest_finalized_block_epoch    9142
        latest_finalized_block_hash     0xc23e1096c628bae3927f95be0edca28c0e98679ec34d01435e04240e404aab41
        latest_finalized_block_height   9142
        latest_finalized_block_time     1734179989
        peer_id                         0x8a95c6C85A4eB294c4dCB764867717C7CDf1f072
        snapshot_block_hash             0x7375fa6eeb782e5534b879ec5243e2bd549223a0ac077cd3c5ae32ec405c34eb
        snapshot_block_height           4397
        syncing                         false
        tip_hash                        0x095078b6da500bb9e0365b84177ec9cfa3ccb43ccc951a654ff412df95c052f0
        explorer_progress_height        9142
```

stv peers

List current neighbours (nodes connected to)

Possible outputs in normal and `-batch` modes

```
$ stv peers
        3 peers
             1  0x8c26b34fa5679d5aef15c9e1bc8a7e7f32debcda
             2  0xf3e2a4bcb781ae96e7d7e6a8f2bde64326fe1acb
             3  0xc52e1be9f3d95f2a8734b3e5c2a6d38fc2b49c41

$ stv -batch peers
        3 0x8c26b34fa5679d5aef15c9e1bc8a7e7f32debcda 0xf3e2a4bcb781ae96e7d7e6a8f2bde64326fe1acb 0xc52e1be9f3d95f2a8734b3e5c2a6d38fc2b49c41
```

Wallet functions

stv new_key

Adds a new address to the local wallet. keys are stored in /home/stv/script4/wallet/keys

Example outputs:

```
$ stv new_key
        Successfully created key: 0x6e5C802Cfc8B2b7c1f559dD372a08a9DA1A2e2b7

$ stv -batch new_key
        address 0x210A046BBCf083EA705fBA7425b348cfCEffa840
```

stv import_key

Imports a private key.

```
$ stv import_key 0102030405060708090a0b0c0d0e0f101112131415161718191a1b1c1d1e1f20
        Successfully imported key: 0x6370eF2f4Db3611D657b90667De398a2Cc2a370C

$ stv -batch import_key 0102030405060708090a0b0c0d0e0f101112131415161718191a1b1c1d1e1f20
        address 0x6370eF2f4Db3611D657b90667De398a2Cc2a370C
```

stv keys

Lists all keys in the local wallet. Example output:

```
$ stv keys
        address 0x8ba4646c50c17b64f9c74a5ac5c4c824f1e92069
        address 0x6370ef2f4db3611d657b90667de398a2cc2a370c
        address 0x4c0f376db9978234c307be858b56e2896548319e
```

stv keysp

Lists all keys in the local wallet along with their private keys.

Example output:

```
$ stv keysp
        address 0x8ba4646c50c17b64f9c74a5ac5c4c824f1e92069 private_key 5c9e6f348db9cdee9a772b74139531dc7fe2f65f53057ce3e85d7dcd685067db
        address 0x6370ef2f4db3611d657b90667de398a2cc2a370c private_key 0102030405060708090a0b0c0d0e0f101112131415161718191a1b1c1d1e1f20
        address 0x4c0f376db9978234c307be858b56e2896548319e private_key 17c2baba90f65eb3f1cec13a362310992091e24d4be6caee9319fa652184707d
```

stv backup

Creates a backup file for the gov and wallet keys. Gives a scp command for retrieving the file from a remote computer in the same LAN, or using internet public address.

```
$ stv backup
        Copying gov key from /home/stv/script4/gov/key/
        Copying wallet keys from /home/stv/script4/wallet/keys/
        Backup file available at /tmp/keys_0x6370eF2f4Db3611D657b90667De398a2Cc2a370C_testnet_user_20241201110412.tgz.
        ================================================================================
        Backup file is ready for download:
        Retrieve from remote computer using either command:
        from LAN: scp root@172.31.13.144:/tmp/keys_0x6370eF2f4Db3611D657b90667De398a2Cc2a370C_testnet_user_20241201110412.tgz .
        from WAN: scp root@18.134.4.199:/tmp/keys_0x6370eF2f4Db3611D657b90667De398a2Cc2a370C_testnet_user_20241201110412.tgz .
        Restore directory (gov key): /home/stv/script4/gov
        Restore directory (wallet keys): /home/stv/script4/wallet
        ================================================================================
```

Or, in batch mode:

```
stv@ip-172-31-13-144:~$ stv -batch backup
        local_file /tmp/keys_0x6370eF2f4Db3611D657b90667De398a2Cc2a370C_testnet_user_20241201111749.tgz
        scp_lan scp root@172.31.13.144:/tmp/keys_0x6370eF2f4Db3611D657b90667De398a2Cc2a370C_testnet_user_20241201111749.tgz .
        scp_wan scp root@18.134.4.199:/tmp/keys_0x6370eF2f4Db3611D657b90667De398a2Cc2a370C_testnet_user_20241201111749.tgz .
        local_restore_dir_gov /home/stv/script4/gov
        local_restore_dir_wallet /home/stv/script4/wallet
```

To retrieve and examine the backup file use the scp command given:

1.- fetch the file"

```
$ scp root@18.134.4.199:/tmp/keys_0x6370eF2f4Db3611D657b90667De398a2Cc2a370C_testnet_user_20241201110421.tgz .
        keys_0x6370eF2f4Db3611D657b90667De398a2Cc2a370C_testnet_user_20241201110421.tgz                      100% 1059    71.1KB/s   00:00
```

2.- uncompress it

```
$ tar -xzf keys_0x6370eF2f4Db3611D657b90667De398a2Cc2a370C_testnet_user_20241201110421.tgz
```

3.- find files:

```
$ cd keys_0x6370eF2f4Db3611D657b90667De398a2Cc2a370C_testnet_user_20241201110421/
        $ find . -type f
        ./wallet/210A046BBCf083EA705fBA7425b348cfCEffa840
        ./wallet/8bA4646C50c17b64F9C74a5aC5c4c824f1E92069
        ./wallet/6e5C802Cfc8B2b7c1f559dD372a08a9DA1A2e2b7
        ./wallet/6370eF2f4Db3611D657b90667De398a2Cc2a370C
        ./wallet/4c0F376db9978234C307Be858B56E2896548319E
        ./gov/6370eF2f4Db3611D657b90667De398a2Cc2a370C
```

Restore locations are provided by the command.

Tokens functions

stv balance

Print SCPT and SPAY amounts available at node_address. Example outputs:

```
$ stv balance
        Address 0x6370eF2f4Db3611D657b90667De398a2Cc2a370C
            SCPT: 0
            SPAY: 0

$ stv -batch balance
        address 0x6370eF2f4Db3611D657b90667De398a2Cc2a370C
        scpt 0
        spay 0
```

stv --address <address> balance

Print SCPT and SPAY amounts available at given address

. Example output:

```
stv --address 803bdedefa468968ebc0067150297cbf12b50b46 balance
        Address 803bdedefa468968ebc0067150297cbf12b50b46
            SCPT: 0
            SPAY: 0
```

Note: Addresses may be printed/collected from user input prefixed by 0x or not indistinguishably.

Faucet (Only testnet)

stv faucet

Provides access to free tokens allowing to test the script network with full capabilities. Example sequence: 1.- check balance:

```
stv:~$ stv balance
        In 0x219B3c8B4f17643B805cb9bc54de452bF32f5CB6
        SCPT: 0
        SPAY: 0
```

2.- invoke faucet:

```
stv:~$ stv faucet
        Tx 0xa799aa31ce630a398cd44870aa2b52f3b9a0ec11596cceb645860d1bf345931e
        Explorer link: https://explorer-testnet.script.tv/txs/0xa799aa31ce630a398cd44870aa2b52f3b9a0ec11596cceb645860d1bf345931e
        Funds will arrive in seconds, check balance.
```

batch mode

```
stv:~$stv -batch faucet
        tx 0xded0a034cca21036f439b670ae5167483f2cd1f28dac47081f5483a8ee571269
        explorer_url https://explorer-testnet.script.tv/txs/0xded0a034cca21036f439b670ae5167483f2cd1f28dac47081f5483a8ee571269
```

3.- check balance again

```
stv:~$ stv balance
        In 0x219B3c8B4f17643B805cb9bc54de452bF32f5CB6
        SCPT: 1000.000000000000000000
        SPAY: 100.000000000000000000
```

The faucet sent us 1000 SCPT and 100 SPAY

stv --recipient <address> faucet

By default the faucet sends tokens to our node address. Specifying –recipient funds will go to the given address instead.

The following sequence of commands illustrates how to fund a new address.

```
stv:~$ stv new_key
             1  Successfully created key: 0xa999E14Cf49f7877fcDd62493eAA4FB4c18bfbf5

        stv:~$ stv --address 0xa999E14Cf49f7877fcDd62493eAA4FB4c18bfbf5 balance
        In 0xa999E14Cf49f7877fcDd62493eAA4FB4c18bfbf5
        SCPT: 0
        SPAY: 0


        stv:~$ stv --recipient 0xa999E14Cf49f7877fcDd62493eAA4FB4c18bfbf5 faucet
        Tx 0xc73324c862c6de916cf17f65e89e4ef2d0fcee3418d6cae6b550f6ff8b2605fb
        Explorer link: https://explorer-testnet.script.tv/txs/0xc73324c862c6de916cf17f65e89e4ef2d0fcee3418d6cae6b550f6ff8b2605fb
        Funds will arrive in seconds, check balance.


        stv:~$ stv --address 0xa999E14Cf49f7877fcDd62493eAA4FB4c18bfbf5 balance
        In 0xa999E14Cf49f7877fcDd62493eAA4FB4c18bfbf5
        SCPT: 1000.000000000000000000
        SPAY: 100.000000000000000000
```

stv --lightning faucet

Request enough funds to acquire a lighning license and stake the node.

```
stv:~$ stv balance
        In 0x219B3c8B4f17643B805cb9bc54de452bF32f5CB6
        SCPT: 3000.000000000000000000
        SPAY: 300.000000000000000000

        stv:~$ stv --lightning faucet
        Faucet for lightning.
        Tx 0x457f3b2f03f3f30c2a81279f0946741c3e126a4f9fa9a62095e39f0061429c2e
        Explorer link: https://explorer-testnet.script.tv/txs/0x457f3b2f03f3f30c2a81279f0946741c3e126a4f9fa9a62095e39f0061429c2e
        Funds will arrive in seconds, check balance.
```

batch mode

```
stv:~$stv -batch --lightning faucet
        tx 0xded0a034cca21036f439b670ae5167483f2cd1f28dac47081f5483a8ee571269
        explorer_url https://explorer-testnet.script.tv/txs/0xded0a034cca21036f439b670ae5167483f2cd1f28dac47081f5483a8ee571269
```

wait 15 seconds... 20K SCPT will be added up to the account

```
stv:~$ stv balance
        In 0x219B3c8B4f17643B805cb9bc54de452bF32f5CB6
        SCPT: 78000.000000000000000000
        SPAY: 400.000000000000000000
```

stv --validator faucet

Request enough funds to acquire a validator license and stake the node.

```
stv:~$ stv balance
        In 0x219B3c8B4f17643B805cb9bc54de452bF32f5CB6
        SCPT: 0
        SPAY: 0

        stv:~$ stv --validator faucet
        Faucet for validator.
        Tx 0x87c35d985163a314b35da6bb4b6df93cdddd280441123e8ac765eb064d3e5505
        Explorer link: https://explorer-testnet.script.tv/txs/0x87c35d985163a314b35da6bb4b6df93cdddd280441123e8ac765eb064d3e5505
        Funds will arrive in seconds, check balance.
```

batch mode

```
stv:~$stv -batch --validator faucet
        tx 0xded0a034cca21036f439b670ae5167483f2cd1f28dac47081f5483a8ee571269
        explorer_url https://explorer-testnet.script.tv/txs/0xded0a034cca21036f439b670ae5167483f2cd1f28dac47081f5483a8ee571269
```

wait 15 seconds... 5M SCPT will be added up to the account

```
stv:~$ stv balance
        In 0x219B3c8B4f17643B805cb9bc54de452bF32f5CB6
        SCPT: 5000000.000000000000000000
        SPAY: 100.000000000000000000
```

Lightning Node (Licensing and Staking)

Lightning nodes form the biggest network. The are responsible for finalizing blocks. They run PoS consensus algorithm where rewards are proportional to the staked amount. The minimum stake to run a licensed lightning node is 20K SCPT.

Lightning License

Use these functions to manage licensing for lightning nodes:

stv --lightning buy_license

Purchase a license. Lightning nodes licenses are free, given at no cost. This is true for mainnet as well.

```
stv:~$ stv --lightning buy_license
        ========================================
        License Purchase
        ========================================
        You are about to purchase a lightning license for the node address: 0x46df18E814a52EfA9ccD0b28E2c1bF6ec36847B7.
        The license fee of 20000 SCPT will be paid from the address: 0x46df18E814a52EfA9ccD0b28E2c1bF6ec36847B7.
        The transaction will be sent to the Script Network node address: 13ea21689a1fbac47308a928fc7e486b21a3e8a7.
        continue? [yes|ctrl-c]
        yes
        Sending 20000 SCPT to 13ea21689a1fbac47308a928fc7e486b21a3e8a7
        Waiting for transaction to be included in a block
        Payment transaction link: https://explorer-testnet.script.tv/txs/0x580b6a2cc7672004e46901223b6684ea49939faa07c36266ee58c6a27acb5891
        License purchased successfully
        Transfer refund in progress. Transaction id: 0xa75db6abe3b113ac35a2155a1ea07e1a450a23648b41723762d3a23c9385976b
        Link to the refund transaction: https://explorer-testnet.script.tv/txs/0xa75db6abe3b113ac35a2155a1ea07e1a450a23648b41723762d3a23c9385976b
```

for batch mode

```
stv:~$ stv --lightning -batch buy_license
        tx 0xe2ddac8956d302328446dca57710b0087ef4a5c456c3ccf6a004553e84e9a440
        explorer_url https://explorer-testnet.script.tv/txs/0xe2ddac8956d302328446dca57710b0087ef4a5c456c3ccf6a004553e84e9a440
        OK License purchased successfully
        txrefund 0xb1a9e5f99e54fb61b866af663b25a130072519d0399fa2353eaeb8dd3385a6ed
        txrefund_explorer_url https://explorer-testnet.script.tv/txs/0xb1a9e5f99e54fb61b866af663b25a130072519d0399fa2353eaeb8dd3385a6ed
```

stv --lightning --referral <other_node_address> buy_license

If someone gave you a referral code use it to purchase a license.

```
stv:~$ stv --lightning --referral 0xEd00e853Dbb617dB4eB6Bb9E5e7326B0ce7f989C buy_license
        Waiting for transaction to be included in a block
        Deleting existing license.json
        License saved as license.json
```

stv redeem

Purchase a license. Lightning nodes licenses are free, given at no cost. This is true for mainnet as well.

```
stv:~$ stv --lightning buy_license
        Waiting for transaction to be included in a block
        License saved as license.json
```

stv --lightning fetch_licenses

retrieve the licensed nodes database. Its a json file placed at `license_db.json`

stv --lightning print_license

Prints on screen a the license featuring LN (Lightning Node) Validity timestamps are in unix time. They are not-expiring perpetual licenses

```
stv:~$ stv --lightning print_license
        ========================================
               LICENSE FOR 0x46df18e814a52efa9ccd0b28e2c1bf6ec36847b7
        ========================================
        | Licensee Address      | 0x46df18e814a52efa9ccd0b28e2c1bf6ec36847b7 |
        | License Type          | LN |
        | Issuer Address        | 0x13ea21689a1fbac47308a928fc7e486b21a3e8a7 |
        | Valid From            | 1733266719 |
        | Valid To              | 4888940319 |
        | License Items         | LN |
        ========================================
        Total Licenses: 1
```

for batch mode

```
stv:~$ stv -batch print_license
        size 1
        issuer 0x13ea21689a1fbac47308a928fc7e486b21a3e8a7
        licensee 0x46df18e814a52efa9ccd0b28e2c1bf6ec36847b7
        from 1733266719
        to 4888940319
        items ["LN"]
        signature 0x00bffab1dad86ff06e20c8b962f4b5f6e66e217547aba08497085a5bdbff184d1fc56ca1239ff4662223207a48537f580035c18627fb2e48548f9915ecec95f101
        #end-of-record
```

stv print_referral

Example message to share with friends.

```
stv:~$ stv print_referral
        Use my node address as referral code:
        0xEd00e853Dbb617dB4eB6Bb9E5e7326B0ce7f989C
        Accepted when purchasing features.
        For instructions on how to install a node visit https://script.tv
```

Staking

On a fresh installation nodes are running but not participating in blockchain build activity, i.e. neither Validating new produced blocks nor running PoS for finalization nor. There two roles are referred to as Validator nodes and Lightning nodes. Once acquired a license for running either role, the next step is to stake the node.

stv --lightning stake

This sequence will obtaing funds and stake a lightning node:

```
stv:~$ stv --lightning faucet
        Faucet for lightning.
        Tx 0x243fef2fdbfcbe00c134d625e0b92caa15e2f980443654025ca304de508d93ca
        Funds will arrive in seconds, check balance.
```

<wait secs>

```
stv:~$ stv balance
        In 0x219B3c8B4f17643B805cb9bc54de452bF32f5CB6
        SCPT: 20000.000000000000000000
        SPAY: 100.000000000000000000

        stv:~$ stv --lightning stake
        Upgrading node to lightning
        Transaction broadcast was successful.
        TxHash: 0xb1a9e5f99e54fb61b866af663b25a130072519d0399fa2353eaeb8dd3385a6ed
        Link to transaction: https://explorer-testnet.script.tv/txs/0xb1a9e5f99e54fb61b866af663b25a130072519d0399fa2353eaeb8dd3385a6ed

        stv:~$ stv balance
        In 0x219B3c8B4f17643B805cb9bc54de452bF32f5CB6
        SCPT: 0
        SPAY: 99.999999000000000000
```

for batch mode

```
stv:~$ stv -batch --lightning stake
        tx 0xb1a9e5f99e54fb61b866af663b25a130072519d0399fa2353eaeb8dd3385a6ed
        explorer_url https://explorer-testnet.script.tv/txs/0xb1a9e5f99e54fb61b866af663b25a130072519d0399fa2353eaeb8dd3385a6ed
```

stv node_status

Provides feedback about the stake Possible outputs:

```
stv:~$ stv node_status
        Node is lightning with 20000.000000000000000000 SCPT staked.
        Node status summary: Node is a lightning.
```

for batch mode

```
stv:~$ stv -batch node_status
        status lightning
        amount 20000.000000000000000000
```

or, if you run both lighning and validator:

```
stv@ip-172-31-8-249:~$ stv node_status
        Node is validator with 1000000.000000000000000000 SCPT staked.
        Node is lightning with 20000.000000000000000000 SCPT staked.
        Node status summary: Node is both validator and lightning.


stv:~$ stv -batch node_status
        status lightning
        amount 20000.000000000000000000
        status validator
        amount 1000000.000000000000000000
```

Validator Node (Licensing and Staking)

Validator nodes form a small network of fast consensus. The are responsible for producing new blocks at a fast pace. They run a non-scalable but very fast pBFT consensus algorithm. The number of seats are limited to 30 (TBC) The minimum stake to run a licensed lightning node is 1M SCPT.

Validator License

Use these functions to manage licensing for validator nodes:

stv print_referral

Share this code and receive rewards. See lightning section above.

```
stv:~$ stv print_referral
        Use my node address as referral code:
        0x219B3c8B4f17643B805cb9bc54de452bF32f5CB6
        Accepted when purchasing features.
        For instructions on how to install a node visit https://script.tv
```

stv --validator buy_license

Purchase a license. Validator nodes licenses cost 4M SCPT.

```
stv:~$ stv --validator buy_license
        ========================================
        License Purchase
        ========================================
        You are about to purchase a validator license for the node address: 0x46df18E814a52EfA9ccD0b28E2c1bF6ec36847B7.
        The license fee of 4000000 SCPT will be paid from the address: 0x46df18E814a52EfA9ccD0b28E2c1bF6ec36847B7.
        The transaction will be sent to the Script Network node address: 13ea21689a1fbac47308a928fc7e486b21a3e8a7.
        continue? [yes|ctrl-c]
        yes
        Sending 4000000 SCPT to 13ea21689a1fbac47308a928fc7e486b21a3e8a7
        Waiting for transaction to be included in a block
        Follow the transaction: https://explorer-testnet.script.tv/txs/0x9b0030e13cfbcf5d7e63764736a3d70ecb8574a12e26b9227b8a5e665b270d10
```

wait 1-2 minutes. Possible output:

```
License purchased successfully
```

for batch mode

```
stv:~$ stv --validator -batch buy_license
        tx 0xe2ddac8956d30232846dca57710b0087ef4a5c456c3ccf6a004553e84e9a440
        explorer_url https://explorer-testnet.script.tv/txs/0xe2ddac8956d302328446dca57710b0087ef4a5c456c3ccf6a004553e84e9a440
        OK License purchased successfully
```

stv --validator --referral <other_node_address> buy_license

Give a referral for extra rewards when purchasing a license

```
stv@ip-172-31-8-249:~$ stv --validator faucet
        Faucet for validator.
        Tx 0x3e7058209ee77f1610478ef279c5efbc090fa92854651650df0aa55cc04807cd
        Funds will arrive in seconds, check balance.
```

wait seconds...

```
stv@ip-172-31-8-249:~$ stv --validator --referral 0xed00e853dbb617db4eb6bb9e5e7326b0ce7f989c buy_license
        Sending 4000000 SCPT to ad5448ab1687f6473715fcb789a2208d14eb9d5e
        Successfully broadcasted transaction: { "hash": "0x77c18d56339526347d743d7c6afbe99cd8dbb5219c3e3a565cae9104a0aa765f", "block": { "ChainID": "testnet", "Epoch": 150, "Height": 150, "Parent": "0xdf129d639
        Waiting for transaction to be included in a block
```

wait for 1-2 minutes.

```
Deleting existing license_db__testnet_user1.json
        License saved as license_db__testnet_user1.json
```

License costed 4M SCPT, we have 1M remaining for staking the node.

```
stv:~$ stv balance
        In 0x219B3c8B4f17643B805cb9bc54de452bF32f5CB6
        SCPT: 1000000.000000000000000000
        SPAY: 99.999999000000000000
```

stv --validator print_license

Print the current validator license on screen.

```
stv:~$ stv print_license
        ========================================
               LICENSE DETAILS VN for 0x46df18e814a52efa9ccd0b28e2c1bf6ec36847b7
        ========================================
        | Licensee Address      | 0x46df18e814a52efa9ccd0b28e2c1bf6ec36847b7 |
        | License Type          | VN            |
        | Issuer Address        | 0x13ea21689a1fbac47308a928fc7e486b21a3e8a7 |
        | Valid From            | 1733301016 |
        | Valid To              | 4888974616 |
        | License Items         | VN |
        ========================================
```

Staking

On a fresh installation nodes are running but not participating in blockchain build activity, i.e. neither Validating new produced blocks nor running PoS for finalization nor. There two roles are referred to as Validator nodes and Lightning nodes. Once acquired a license for running either role, the next step is to stake the node.

stv --validator stake

To stake a validator node follow this example including how to obtain funds from the faucet, but not needed it you come from buying license just above.

```
stv:~$ stv --validator faucet
        Faucet for validator.
        Tx 0x04b2bb44535630ea705c2e126722e58c00cc640aa4bf732b5647a3509ac6a54c
        Funds will arrive in seconds, check balance.

        stv:~$ stv balance
        In 0x219B3c8B4f17643B805cb9bc54de452bF32f5CB6
        SCPT: 1000000.000000000000000000
        SPAY: 199.999999000000000000

        stv:~$ stv --validator stake
        Upgrading node to validator
        Successfully broadcasted transaction.

        stv:~$ stv balance
        In 0x219B3c8B4f17643B805cb9bc54de452bF32f5CB6
        SCPT: 0
        SPAY: 199.999996000000000000
```

for batch mode

```
stv:~$ stv --validator -batch stake
        tx 0x49bcad047f26e30e7823d615d7b2d22999d04bbfd70a97f661e59e5b9baf6bb0
        explorer_url https://explorer-testnet.script.tv/txs/0x49bcad047f26e30e7823d615d7b2d22999d04bbfd70a97f661e59e5b9baf6bb0
```

Staking the validator has consumed our remaining 1M SCPT.

stv node_status

Provides feedback about the stake

```
stv:~$ stv node_status
        Node is validator with 1000000.000000000000000000 SCPT staked.
        Node status summary: Node is a validator.
```

for batch mode

```
stv:~$ stv node_status
        status validator
        amount 1001000.000000000000000000
```

stv transfer

Transfer tokens to other addresses

```
stv --scpt 100 --to 0x5e4Fc7c1037F62a7726d389B6b7765a240F026E6 transfer
        SENDER ACCOUNT
        ==============
        Address: 0x46df18E814a52EfA9ccD0b28E2c1bF6ec36847B7
        Balance:
          SCPT: 1000.000000000000000000
          SPAY: 100.000000000000000000

        RECIPIENT ACCOUNT
        =================
        Address: 0x5e4Fc7c1037F62a7726d389B6b7765a240F026E6
        Amounts sent:
          SCPT: 100
          SPAY: 0

        COST
        ====
        Fees paid by sender: 0 SPAY
        Total cost:
          SCPT: 100
          SPAY: 0

        Confirm? [yes|{ctrl-c}]: yes
        Transaction hash: 0x8a4a470f3def9a6b6994f01d45a408f62e0a11c060d58043aa7b0290c6af704c
        Follow the transaction: https://explorer-testnet.script.tv/txs/0x8a4a470f3def9a6b6994f01d45a408f62e0a11c060d58043aa7b0290c6af704c
```

for batch mode

```
stv -batch --scpt 100 --to 0x5e4Fc7c1037F62a7726d389B6b7765a240F026E6 transfer
        from_addr 0x46df18E814a52EfA9ccD0b28E2c1bF6ec36847B7
        to_addr 0x5e4Fc7c1037F62a7726d389B6b7765a240F026E6
        amount_scpt 100
        amount_spay 0
        tx 0x2faee259f5090d742df44e46ed34bb2a300fae7480f24b112a07809a46e74c11
        explorer_url https://explorer-testnet.script.tv/txs/0x2faee259f5090d742df44e46ed34bb2a300fae7480f24b112a07809a46e74c11
```

Appendix I - nodeops automation

The stv program can be integrated in higher level automation scripts.

This is an example of a possible sequence for instantiating a new node in a given VM, with a given node key, and such key managed by a company wallet:

```
1.- create VM with debian 12 OS, save its IP address
        2 - Import a given private key or generate it.
            stv import_key <node_key>     #if you generated the key elsewhere.
            stv new key
            stv keysp        #save private key and use it as node_key, save the node_address
        3.- Transfer 20K SCPT some SPAY for tx fees to this new address
            stv transfer --to <node_address> --scpt 20000
        4.- create, transfer and execute a bootstrap script
```

```
4.- scp bootstrap root@<IP_Address>:~/
5.- ssh root@<IP_Address> "chmod+ x bootstrap; ./bootstrap <node_key>"
```

go to https://download.script.tv/download-install, copy the 1liner A possible bootstrap script would be:

```
#/bin/bash

        node_key=$1
        oneliner="<paste the 1liner here>"
        stv="bin/stvtool -batch"   #define stv command for batch/non-interactive mode

        #append args to 1liner
        myoneliner="${oneliner} -batch --node_key ${node_key}"

        #Run the oneliner as root
        ${myoneliner}

        wait 300 #let node startup

        # Create a temporary file
        TMP_CMD_FILE=$(mktemp)

        # Write commands to the temporary file
        cat << EOF > "$TMP_CMD_FILE"
        cd /home/stv
        # Commands for the stv user
        #= custom commands, as user stv ============================

        balance="$($stv balance)"
        echo $balance

        #and so on with any command
        $stv buy_license
        $stv --lightning stake
        #...

        #===========================================================
        EOF

        # Ensure the file has the correct permissions
        chmod 700 "$TMP_CMD_FILE"

        # Run the commands as the stv user
        su - stv -c "bash $TMP_CMD_FILE"

        # Clean up the temporary file
        rm -f "$TMP_CMD_FILE"

        exit 0
```

Appendix II - Software updates

The system is programmed to check for updates once a day by default.

disabling automatic updates

If you wish to disable automatic updates type the following command as user root:

```
mv /etc/cron.d/script_tv_update /root
        systemctl restart cron
```

To enable it again reverse bring the file back in the cron.d directory and restart the crond daemon:

```
mv /root/script_tv_update /etc/cron.d/
        systemctl restart cron
```

This crontab file is programmed to execute periodically the program `script_tv__update.sh`

```
stv@ip-172-31-13-144:~$ cat /etc/cron.d/script_tv_update
        3 8 * * * root timeout 3600 /bin/bash /usr/local/bin/script_tv__update.sh > /dev/null 2>&1
```

You can run (as root) this program manually anytime for installing updates.

```
script_tv__update.sh
```