#HORIZEN STAKE VERIFICATION TOOL WINDOWS INSTRUCTIONS

This tool is used to verify the ownership of a ZEN transparent address used for staking Horizen Super and Secure Nodes.

Verified addresses allow:

\* up to five alternate 'payto' addresses

\* multiple stake amounts in one address

A stake address is verified by sending a deterministically derived amount of ZEN from the stake address back to the same address and submitting the resulting transaction id and the information used to derive the amount back to the node tracking server.

The tool consists of two command line applications:

\* staketool

\* creates verification requests and ZEN transactions

\* sends signed ZEN transactions to the blockchain and verification requests to the servers

\* signtxtool

\* signs ZEN transactions using the private key of the stake address

Stand alone binaries for Windows can be downloaded under [releases](<https://github.com/HorizenOfficial/staketool/releases/>).

After Downloads extract the zip file **staketool-windows-x.x.x.zip** (the “x.x.x” representing the tool version number) to any folder where you will find staketool.exe and signtxtool. Copy the file absolute file path of the aforementioned tools and add the path to your *Environment Variables* settings by first either right clicking start button or using shortcut **WINDOWS KEY + X**, and then go to **Settings** from there click on **Advanced System Settings** and a new window will appear on there click on Environmental variables after that look for a variable under system variables called path and depending on your version of windows either concatenate the path for the staketool and signtxtoool by adding “;” without the quotes to the end of the path variables that exist or in the case of latest versions of windows click on the path variable to highlight it and hit Edit>New>Okay>Okay. Another way of accomplishing the previous action would be to open PowerShell with Administrative privileges and type the following line of commands.

$env:Path += ";Path of Staketool and Signtxtool" after that in a new Command Shell window both the staketool and signtxtool can be accessed from any location on the PC.

If you prefer to run the CLI apps under nodejs (>=14) you can do so by cloning the repository and running `npm ci`. After that you can run the tools by using `node staketool.js` and `node signtxtool.js`.

What You Will Need

\* The ZEN address where the stake amount will reside, a small amount to create the verification transaction is needed in this address (<0.1 ZEN). No Super or Secure Nodes are allowed to be registered with this address prior to verification, it has to be a new stake address. The same stake address cannot be verified on both tracking systems.

\* One to five 'payto' addresses. A percentage of the node earnings may be split between addresses.

\* The private key of the stake address, unless you prefer to send the verification transaction from a wallet like [Sphere by Horizen](https://github.com/HorizenOfficial/Sphere\_by\_Horizen) or use the zen-cli command line.

\* A Horizen Super or Secure Node API sub key. This sub key is provided by your Node hosting provider or, if you host your own nodes, you can create it on the [Secure Node API Settings](https://securenodes.zensystem.io/settings/) or [Super Node API Settings](https://supernodes.zensystem.io/settings/) page.

otes:

\* Only transparent addresses are supported as stake addresses (not private z-addresses).

\* Staketool does not yet support creating multisignature transactions verifying an address, though it is possible to verify multisignature addresses. Contact Support for instructions.

### General Steps

The Stake Verification Tool has three general steps to complete the verification process:

1. Create a stake verification request and a ZEN transaction for the specified amount.

2. Sign the transaction. This step can be done offline.

3. Send the transaction to the blockchain and the verification request to the desired Node tracking servers.

### Stake Tool Example

The tool has the flexibility to complete the verification process in multiple ways. The ZEN may also be sent from the stake address using a wallet or the zen-cli command line. Detailed usage help for each command is available in [USAGE.md](USAGE.md).

The most straightforward process is to use the tool with the default options.

#### Step 1. Create the Verification Request and Transaction

Command line:

`staketool createstakeverification --stake=ztpwi9FUYVjPRFEXsqTZ5zwvtUZCdaUNYy4 --payaddress='[{"address":"ztpwi9FUYVjPRFEXsqTZ5zwvtUZCdaUNYy4","pct":60.00},{"address":"ztjSb53TJSLEG3VBqoBiTiZLkken6nL4ZJN","pct":40.00}]' --system=super`

The `--payaddress` argument data must be valid JSON and consist of one to five 'payto' addresses with a percentage split for each. All pct must add up to 100. Depending on your operating system and shell, different forms of escaping/encapsulating in quotes have to be used to provide the `--payaddress` data. An `"address":` in `--payaddress` may be the same as the `--stake`.

The `--system` argument is required. Valid values are `super, secure, or testnet`

#### Step 2. Sign the ZEN Transaction

Command line:

`signtxtool signverificationtransaction --privkey=cQ3iBWAa2BwNCAjBWPMRryuib3QbfYwbfhbB8dpJnKShowfXp43x`

This step may be completed on an air gapped system by copying the `.\verificationfiles` folder created in Step 1. Copy the files back to the first system and replace the originals once the transaction is signed.

The private key can be provided using the `--privkey` argument, or by exporting the `PRIVKEY=privateKeyWIF` environment variable. If neither is used, the tool will prompt to input the private key.

#### Step 3. Send the ZEN Transaction and the Verification Request

Command line:

`staketool sendtxandstakeverification --apikey=xn571c0cf42af2jcbb8dab2fafb973bfd14a2c0dbd --system=super`

This step should result in a status message from the Node tracking server. The verification request is confirmed after five blocks.

#### Assigning a Stake Address to a Node

Once a stake address is verified it can be used to register a new Super or Secure Node or to reconfigure an existing Node. The `"email":`, `"stakeaddr":` and `"category":` fields in the node's `config.json` file have to match the values of the API sub key that was used to submit the verification request. For detailed instructions on changing a stake address see https://horizenofficial.atlassian.net/wiki/spaces/ZEN/pages/136872049/Change+Staking+Address.

#### Changing 'payto' Information of a Verified Address

Rerun Steps 1-3 with the same stake address but provide different 'payto' addresses or percentages. After successful verification the new 'payto' addresses will be used starting from the next payment period.

### Additional Commands

Detailed usage help for each of these additional commands is available in [USAGE.md](USAGE.md).

#### List Existing Stakes - Retrieve the current list of stakes and their status from the tracking servers.

Command line: `staketool liststakes --apikey=xn571c0cf42af2jcbb8dab2fafb973bfd14a2c0dbd --system=super`

#### Get Balance - Check the current balance of any zen address.

Command line: `staketool getbalance --stake=ztpwi9FUYVjPRFEXsqTZ5zwvtUZCdaUNYy4 --system=super`

#### Cancel Request - Cancel a confirming or verified request.

Command line: `staketool cancelstakeverification --idstake=201 --apikey=xn571c0cf42af2jcbb8dab2fafb973bfd14a2c0dbd --system=super`

The stake id for --idstake can be found using the 'liststakes' command

#### Derive Addresses and Keys From Seed Phrase

Command line: `signtxtool keysfromseed --seed='my seed phrase'`

The signtx tool can list your addresses and private keys from your seed phrase. CAUTION: only perform this action on a secure system.

### Development

```

git clone https://github.com/HorizenOfficial/staketool.git

cd staketool

npm ci

npm run test # basic linting and audit checks

npm run makebins # package stand alone binaries for all platforms, output in ./bin

```

#### Testing

For development and testing of integrations, staketool can be used with the testnet tracking servers at https://securenodes.testnet.zensystem.io. To do so please use testnet addresses and private keys, and use the `--system=testnet` switch with staketool and signtxtool.

Testnet ZEN can be requested at https://testnet.getzen.cash, testnet addresses and private keys can be generated at https://myzenwallet.io by selecting 'testnet' in Settings.

#### Internals

The tool deterministically derives the amount in satoshis to send in the verification transaction by taking part of a sha256 hash of a JSON object comprising the stake and 'payto' addresses.

The reference implementation of creating the `hashobj` can be found in [helpers/steps.js#L29-L72](https://github.com/HorizenOfficial/staketool/blob/master/helpers/steps.js#L29-L72)

The reference implementation of the algorithm deriving the amount in satoshis from the `hashobj` can be found in [helpers/utils.js#L17-L30](https://github.com/HorizenOfficial/staketool/blob/master/helpers/utils.js#L17-L30)

The `hashobj` is submitted to the tracking servers, the amount in satoshis is calculated again and compared to the on-chain verification transaction. Only the last 8 digits of the satoshi value are compared. To make preserving of JSON element order easier `hashobj` is base64 encoded, the reference implementation sending a verification POST request to the tracking server API at https://supernodes1.eu.zensystem.io/api/stake/verify or https://securenodes1.eu.zensystem.io/api/stake/verify can be found in [helpers/steps.js#L221-L241](https://github.com/HorizenOfficial/staketool/blob/master/helpers/steps.js#L221-L241) and [helpers/utils.js#L182-L215](https://github.com/HorizenOfficial/staketool/blob/master/helpers/utils.js#L182-L215).