Newbie guide ZHCASH

The plan, to be sure, was excellent:
simple and clear, it is better not to think up.
He had only one drawback:
it was completely unknown
how to carry it out.

Alice in Wonderland

If I had this guide
then I would have finished smart a month earlier

Author

DISCLAIMER

This version of the guide is unofficial and made in arthouse style.
Revision 1.1 as of 00/22/2022

This guide describes the nuances of writing smarts for zh, possible mistakes of novice developers, how to work with a node through the terminal and python, launching your local test blockchain, the nuances of a web wallet, possible RPC errors and installing a node via ssh on an ubunta server

Definitions:

1) zh is the 5th generation blockchain ZHCASH. The result of crossing (love) bitcoin and ether, which has undergone genetic modificationhttps://zh.cash/ . Pronounced as "zh". It is the opposite of xs.

2) zx, shekel - the main coin ZHC of the ZHCASH blockchain.

3) Zeroshka is an analogue of Satoshi in bitcoin. Equal to 10^{-eight}sx

4) elevator - token of the LIFT project

5) ZRC20 - analogue of ERC20 and QRC20 (QTUM) standards

7) QTUM is the parent of zhhttps://qtum.org/en . 2 years older. The most promising blockchain in the world (from the Chinese) was taken at the time of 2019 and significantly improved to the best in the world zh. There is good support in the telegram, where they will quickly answer any question.

8) Console - command line in the zh wallet terminal or a program with an interface in the zerohour-cli command line. This is not a sitehttps://zhcash.org/ .

Used to enter commands and interact with the blockchain. API is on site https://zh.cash/docs/en/ZHCash-RPC-API/

8*) Web Console - Website https://zhcash.org/ .

9) hex — real wallet address in HEX format. It is according to it that tokens and shekels are accrued when interacting through a smart contract. Received when you enter the command
gethexaddress ZEFnGiHuwdSthnBA3cvAgPPFhhAKKqXQna
to the console. It turns out 184eb41e30b0d5974df3d1b2429fbdf728222a4c
It's almost an Ethereum address, except it doesn't have a 0x in front of it. It cannot be used in the smart code (it does not compile), when adding 0x at the beginning it is not perceived in zh as a wallet. Use exclusively this type of wallet (without 0x at the beginning), but do not use it directly in the smart code itself.

10) Remix - on-air environment for smart development https://remix.ethereum.org/

12) Broken - a smart (or transaction) that did not have enough gas and did not integrate into the blockchain. Displayed in black in the explorer (zeroscan). For one beaten they give two unbeaten or vice versa.

12) Explorer (Zeroscan) - https://zeroscan.io its api is described at the end

13) Smart - smart contract on solidity

14) Accrue (acquired) - receive native coins from the test network, interest on tokens or receive a reward for block mining. The main condition for this to happen is relaxed

15) Drop - centralized distribution of tokens or shekels at the beginning of the smart launch. It is short for air drop.

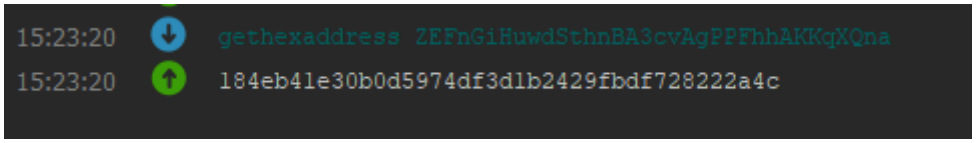
16) UTXO (Unspent Transaction Output) - output of unspent transactions https://2bitcoins.ru/chto-takoe-utxo-i-zachem-on-nuzhen/
When sending 1 sx from a balance of 10 sx, the entire amount is sent to the address, and the change of 9 sx is returned in the next block. It does not allow many transactions to be made in the blockchain, but it also increases the security of the blockchain. Used in bitcoin. The account system is used on the air. Comparison of these systems is described here https://russianblogs.com/article/24511021659/

Smart development under zh

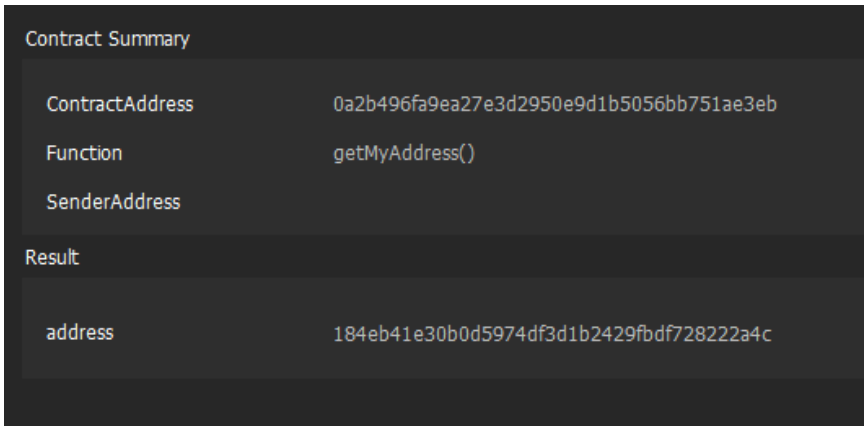
It is assumed that the reader already has an initial level of solidity knowledge, which he can learn for example herehttps://inaword.ru/smart-kontrakty/ or https://www.tutorialspoint.com/solidity/solidity_variables.htm

The zh smart system is identical to Etherium, but there are some nuances. The address in zh is the same Ether address, but without the 0x. But this type of address cannot be specified directly in the remix code, so if we want to pass the value of the address (For example, the first user who will receive a million tokens), then we can do this when creating the contract. In this case, the address should be written in hex format.

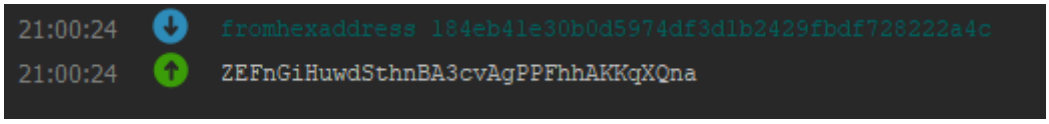
You can get hex through the gethexaddress console (look at the first line on the screen below)



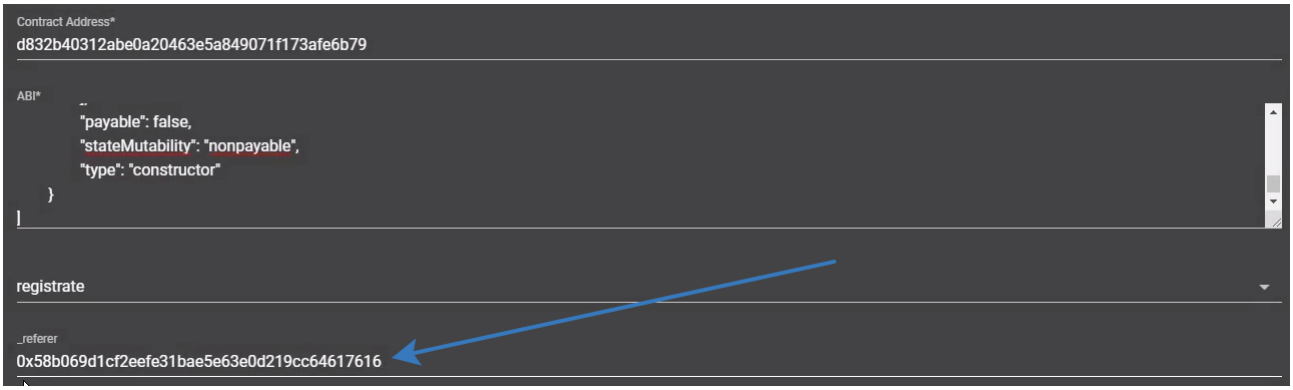
Using the function of my smart to get my wallet address, we note that it will also be in hex format



You can overtake hex to the classic view with the command (take a closer look at the first line on the screen below)

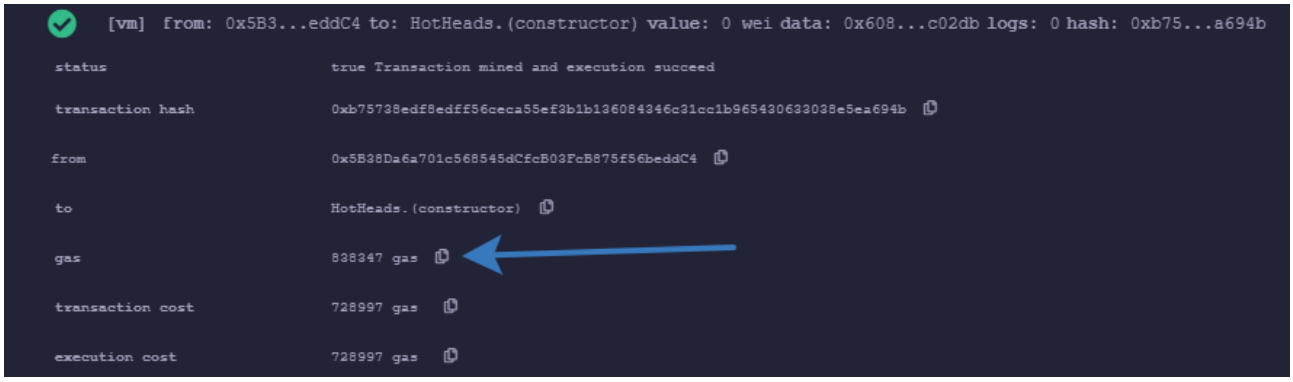


In a web wallet, by the way, when sending any wallet to smart in the form of data, in addition to the hex format, you also need to add 0x as in the good old classic

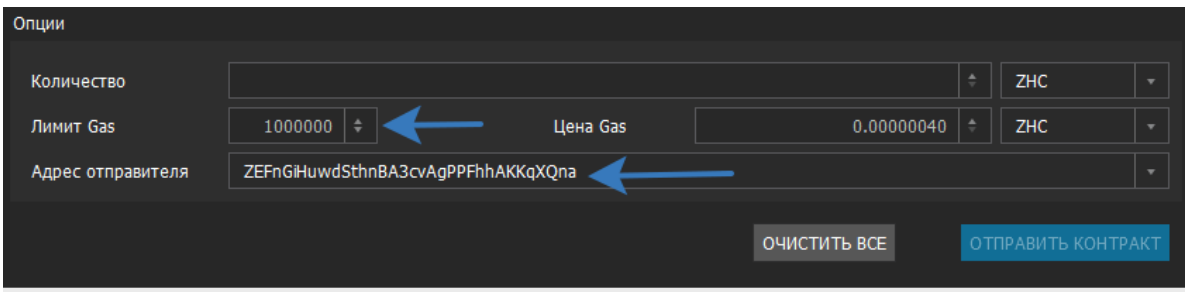


With aidrop, you should not put more than 50 transactions in one block. Otherwise, the entire amount on the balance sheet is spent. 20 transactions are safe.

If the contract turns out to be broken (after it is unloaded, there is no “mined” icon in transactions and it will be displayed black on the zeroscan in the smart block), then you should increase the gas. It is recommended to bet 20% more than shown in the remix

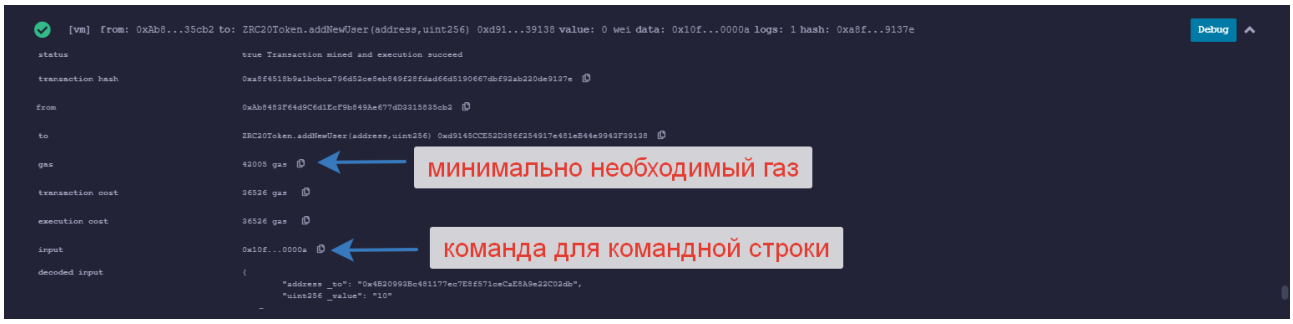


Also indicate from which wallet the smart will be created



The normal situations are shown below.

You can find out the required gas in the remix by expanding the transaction data. But when trying to send the transaction shown in the picture below, it failed with a gas of 50,000, but it did with a gas of 100,000. Recommended gas for any transactions at zh 250000, at qtum 100000.



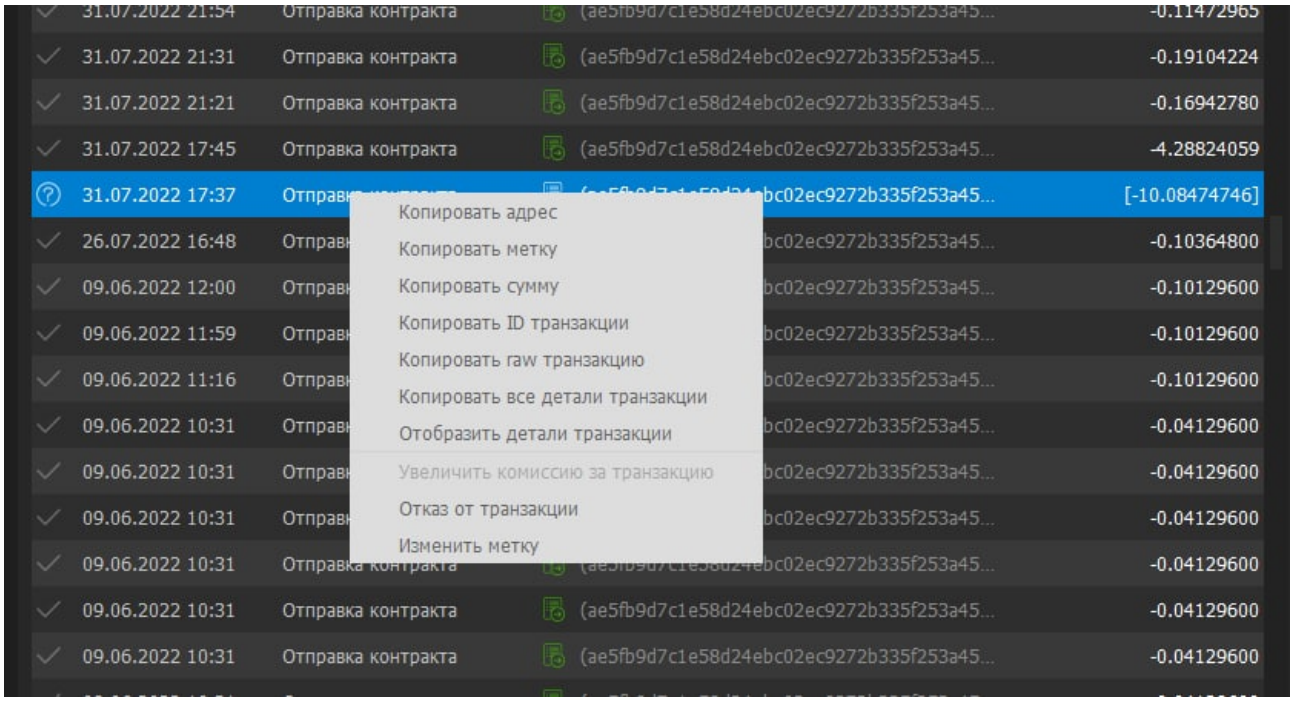
Also there you can find out which command (command code) is being executed. This is necessary to interact with the blockchain through the console.

Zh supports the latest version of solidity (may not work correctly), but it is recommended to use version 0.4.18 (because ABI in later versions does not support accessing smart via call (call)). In the first contracts, I used version 0.4.18, as in the standard qtum QRC20 Token example https://docs.qtum.site/en/QRC20-Token-Introduce.html , because it did not change the standard value of the gas. And when compiling on versions 0.7, even the standard example of smart beats if you leave the gas at 250000.

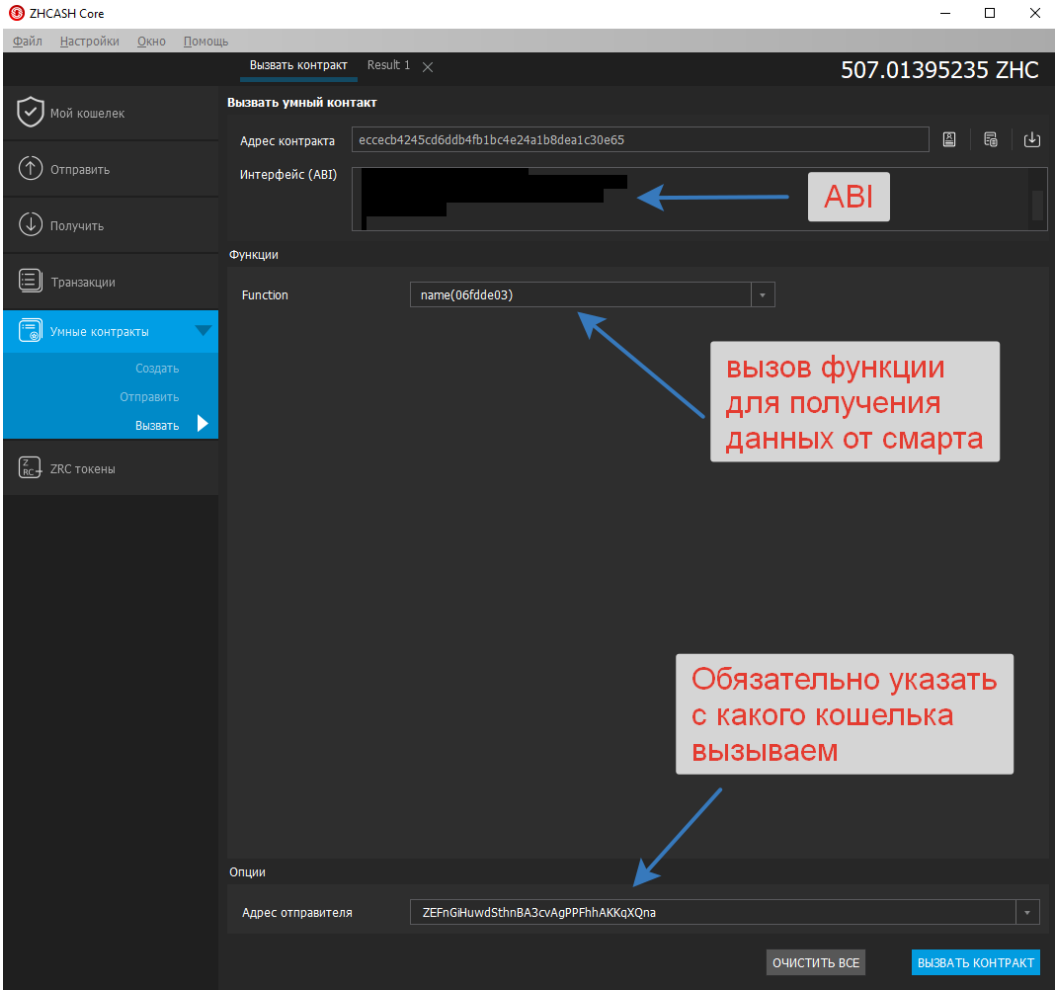
It is recommended to download smarts and send data (sendtocontract) to a smart from a wallet with a balance of no more than 1000 shekels, because. there is a high probability of breaking through the entire balance. The author has dealt with this many times. This rule does not apply to calling contract functions (callcontract) and single sending of tokens to anyone. It was noticed that when trying to cram more than 30 transactions into one block from one wallet, a strong write-off of shekels from the balance (from 1 thousand to 400 thousand) begins. So, you can drain the entire balance of the node (1 million or more easily) to the drop by sending out tokens to 60 users in one

block. As a result, the author made a drop mailing to 20 users in one block (with a delay of 10 minutes). It is sent to 200 users per hour, which is acceptable. For 4 hours, the drop was full of everyone. Still, there are ways to get around this limitation and credit tokens to thousands of addresses instantly (by breaking your wallet into many small ones of one shekel, or using the so-called "batch transactions" sendrawtransactionhttps://zh.cash/docs/en/ZHCash-RPC-API/#sendrawtransaction), or write a smart that will receive sx and make shipments to the specified addresses (smart can do a batch transaction), but we leave this exercise to the reader.

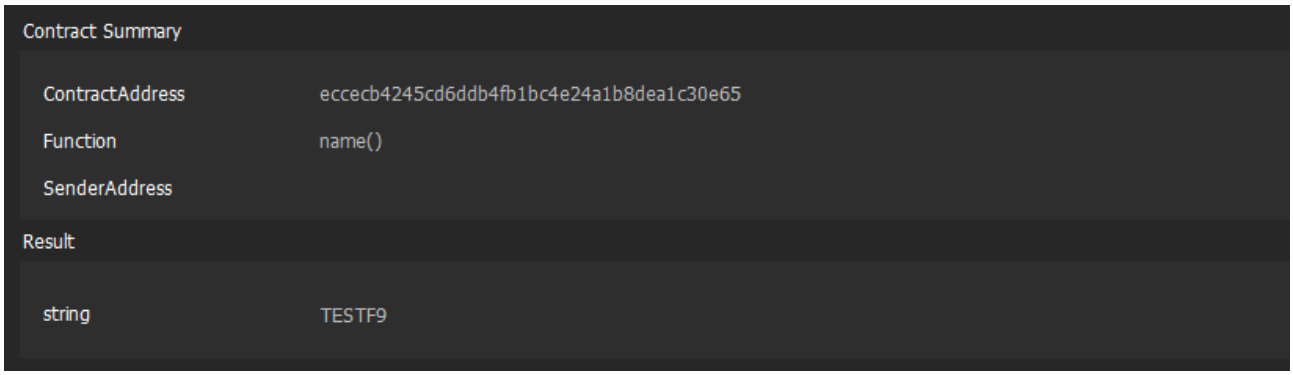
This is due to the fact that change according to the UTXO algorithm must return in the next block, and the entire balance may be used up. Due to the curvature of UTXO (or my programs), it may no longer be displayed in the general balance of the wallet (the transaction may freeze) So 400 thousand shekels were lost in 5 minutes. You can cancel a frozen transaction at any time (even after several months) by clicking on "Cancellation of the transaction" and after cancellation, the lost amount is returned to the balance. This transaction is displayed in question (instead of a checkmark on the left).



Consider how to exchange information with smart. Let's first analyze how to interact with smart through the graphical interface of the wallet. Then we will look at how to do this through the command line.

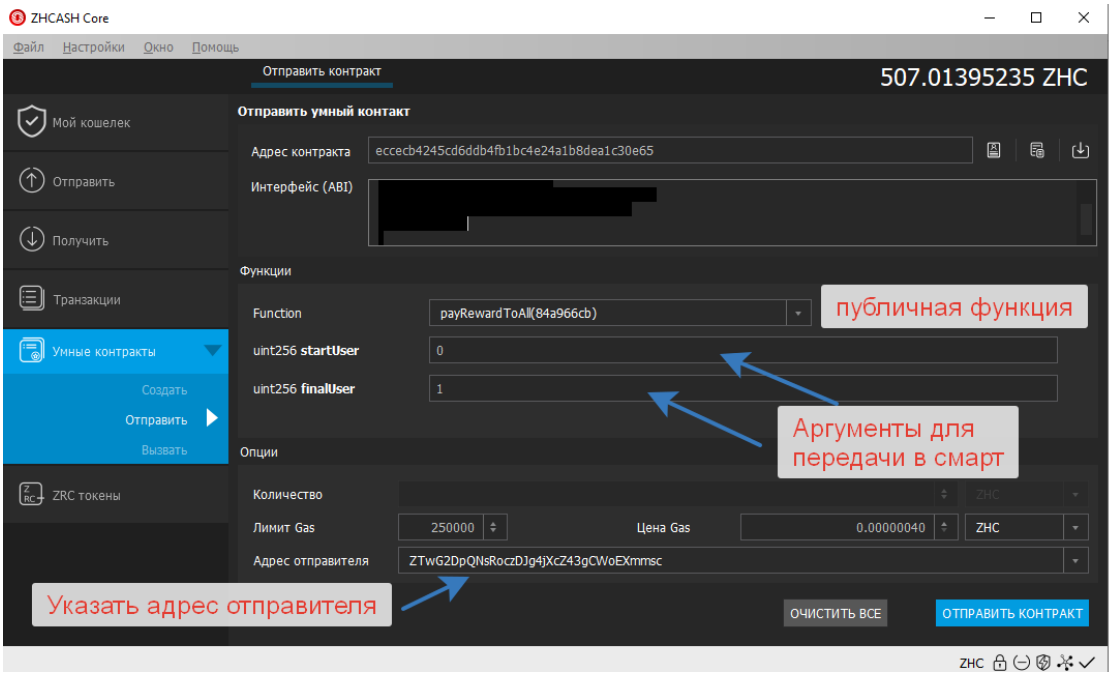


If we want to get data, then we should use the "Call" tab. We get the following result.

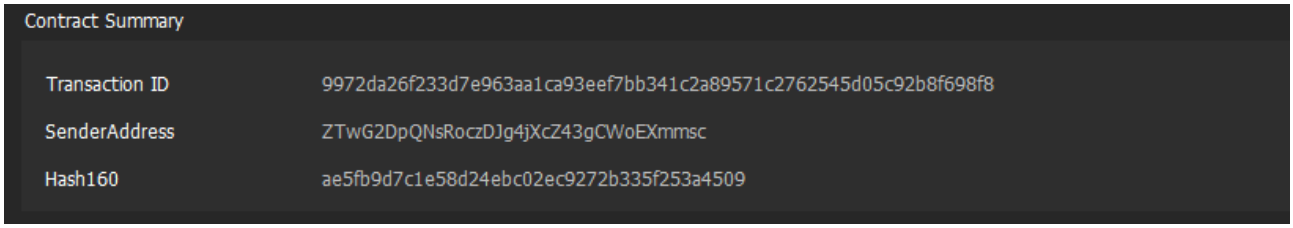


So we can get any value of a public variable or the execution of an external (external view) function.

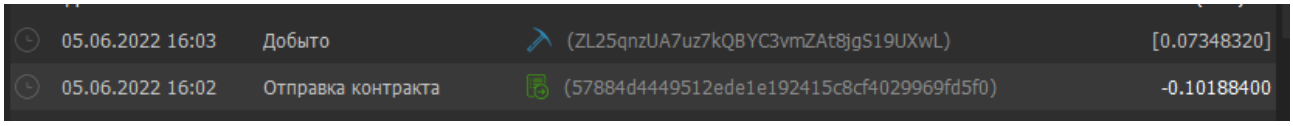
If we want to send data, then we should use the "Submit" tab



If we want to pay rewards to all users, then we should also pay in chunks of 20 transactions per block. And put a higher gas value. After such an inversion, we get the following.



After waiting for a new block and going to "Transactions", we will notice that the "Produced" icon will appear



This means that the data was sent successfully.

Now consider the interaction with the blockchain through the console.

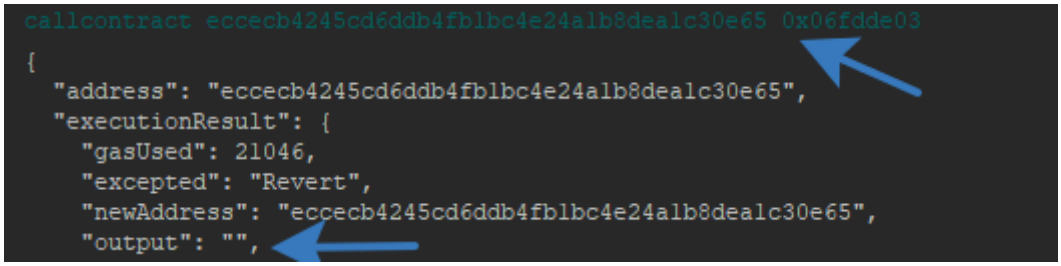
There are two commands for this: callcontract to receive data and sendtocontract to send data to the smart. Below is an example of usage.

sendtocontract efa851007505513c91c0f80e3e9f571a544d5245 a9059cbb000000000000000000000001c96419179e5d6e2cb98b0b2c a658daff6d12cb50000000000000000000000000000000000000000000000000000000000000003b9aca00 0 100000 0.0000004 ZTwG2DpQNsRoczDJg4jXcZ43gCWoEXmmsc

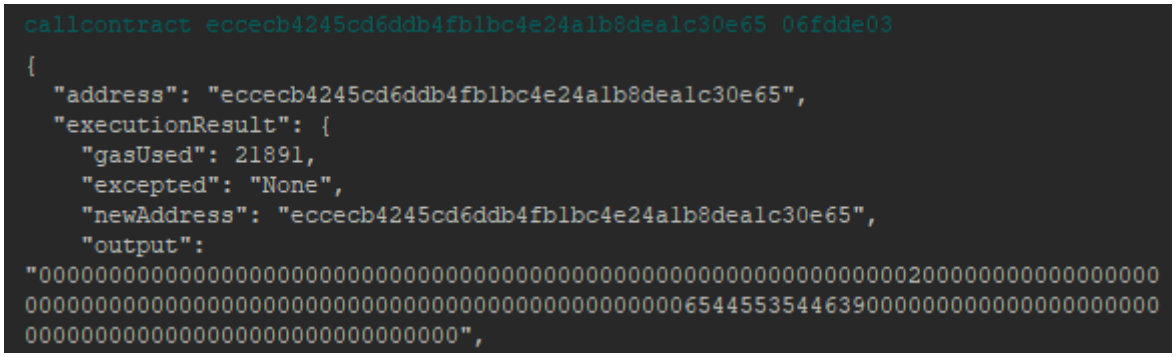
callcontract eccecb4245cd6ddb4fb1bc4e24a1b8dea1c30e65 06fdde03

The command for the transaction through the console can be obtained from the remix

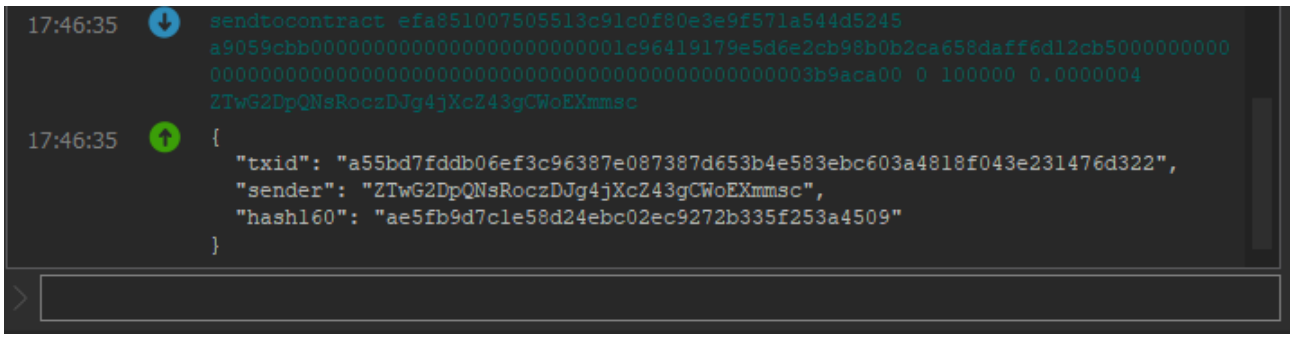
Do not write 0x in the command name. In zh, this is wrong. So you won't get anything. You should write what comes after 0x.



This is how it would be correct:

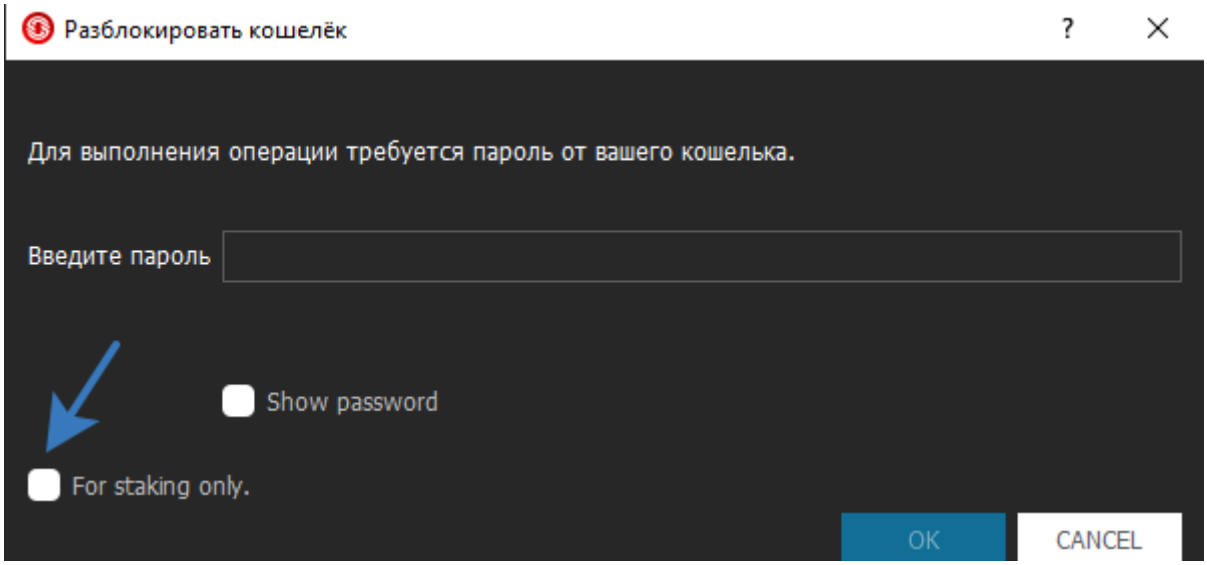


We will get some data, which can then be converted to a string. Below is a screenshot for sendtocontract

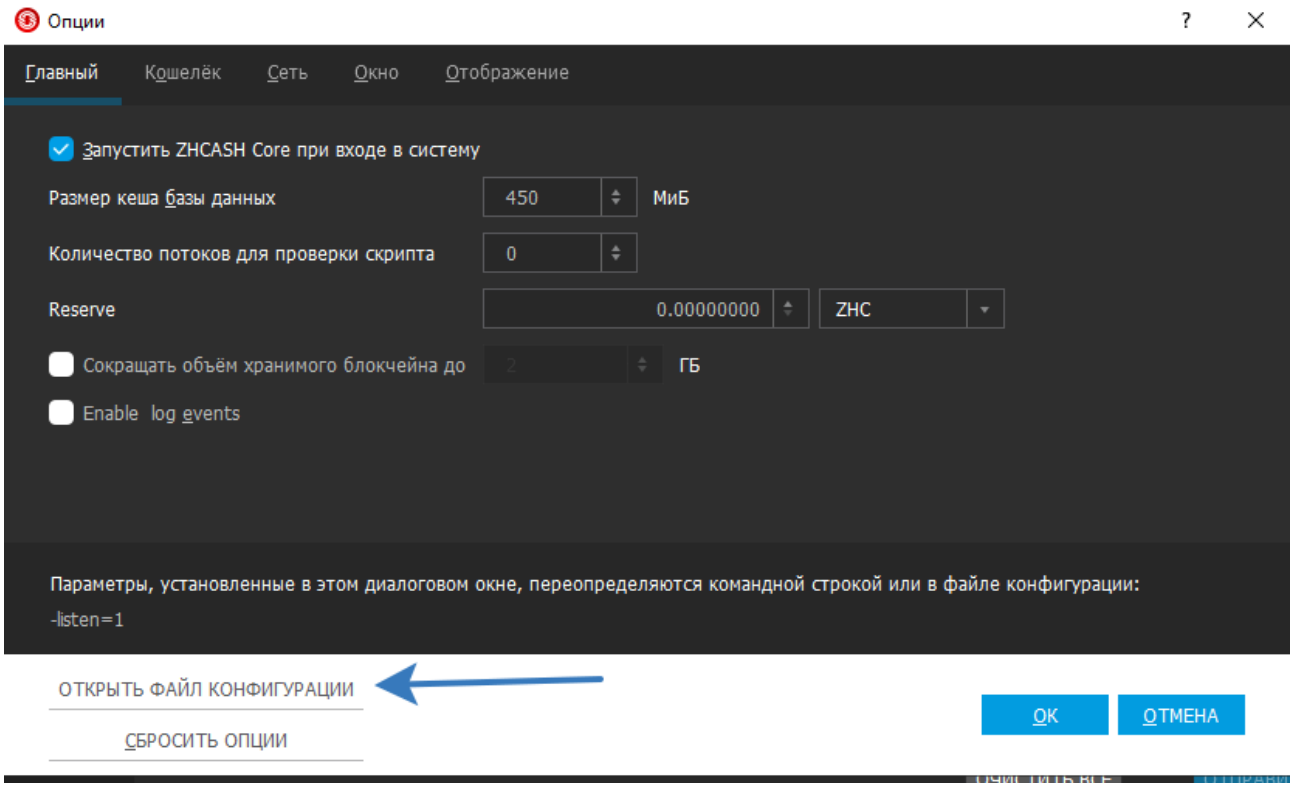


In order to interact via the cmd command line, you need to download the console (server) version of the wallet

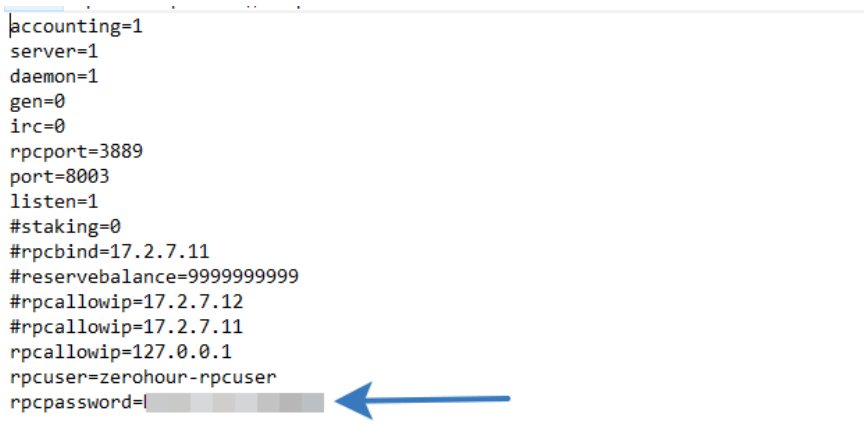
Then unlock the wallet and uncheck “For staking only”



Add config file in options



And save the following text, where in the last argument rpcpassword set your wallet password



accounting=1
server=1
daemon=1
gen=0
irc=0
rpcport=3889
port=8003
listen=1
# staking=0
# rpcbind=17.2.7.11
# reservebalance=9999999999
# rpcallowip=17.2.7.12
# rpcallowip=17.2.7.11
rpcallowip=127.0.0.1
rpcuser=zerohour-rpcuser
rpcpassword=1123581321

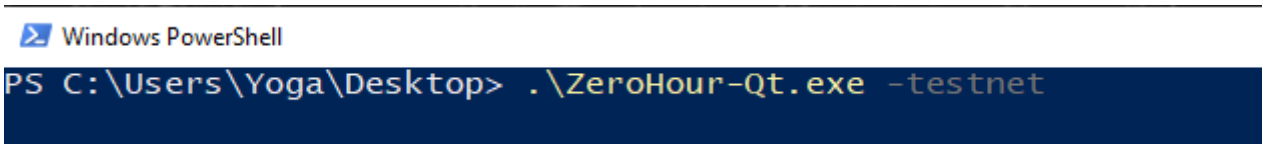
After these manipulations, you can interact with the blockchain through the command line, which allows you to write python scripts to automate some actions with the blockchain. For example, to organize a drop of tokens, send transactions, create a million test tokens, spam, hacking, and the like. Below is how you can call the callcontract function via cmd

```
PS C:\Users\Yoga\Desktop\ZHCASH_oracle> .\zerohour-cli.exe callcontract eccecb4245cd6ddb4fb1bc4e24a1b8dea1c30e65 06
fdde03
{
  "address": "eccecb4245cd6ddb4fb1bc4e24a1b8dea1c30e65",
  "executionResult": {
    "gasUsed": 21891,
    "excepted": "None",
    "newAddress": "eccecb4245cd6ddb4fb1bc4e24a1b8dea1c30e65",
    "output": "0000000000000000000000000000000000000000000000000000000000000020000000000000000000000000000000000000
0000000000000000000000006544553544639000000000000000000000000000000000000000000000000000000000000000000",
```

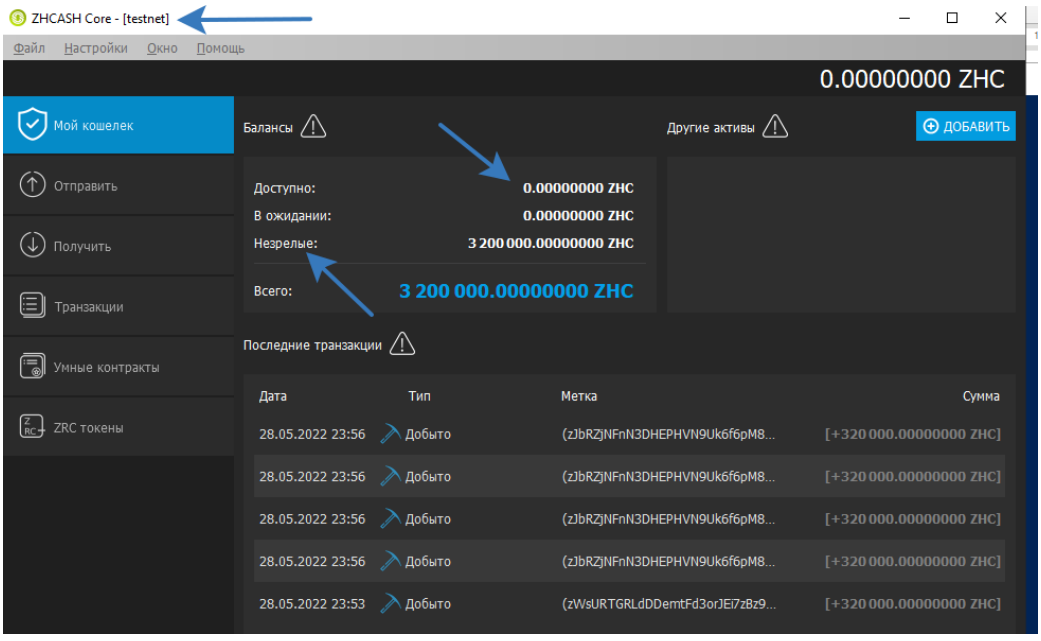
Smart testing

Three approaches can be used to test the smart.

1) Run your zh wallet in testnet mode (with the -testnet switch via the command line).



In this method, you organize a local blockchain from block zero, where you still need to generate 500 blocks in order to get test shekels. Until you do this, all shekels will be immature and you will not be able to pay for the creation of a smart (

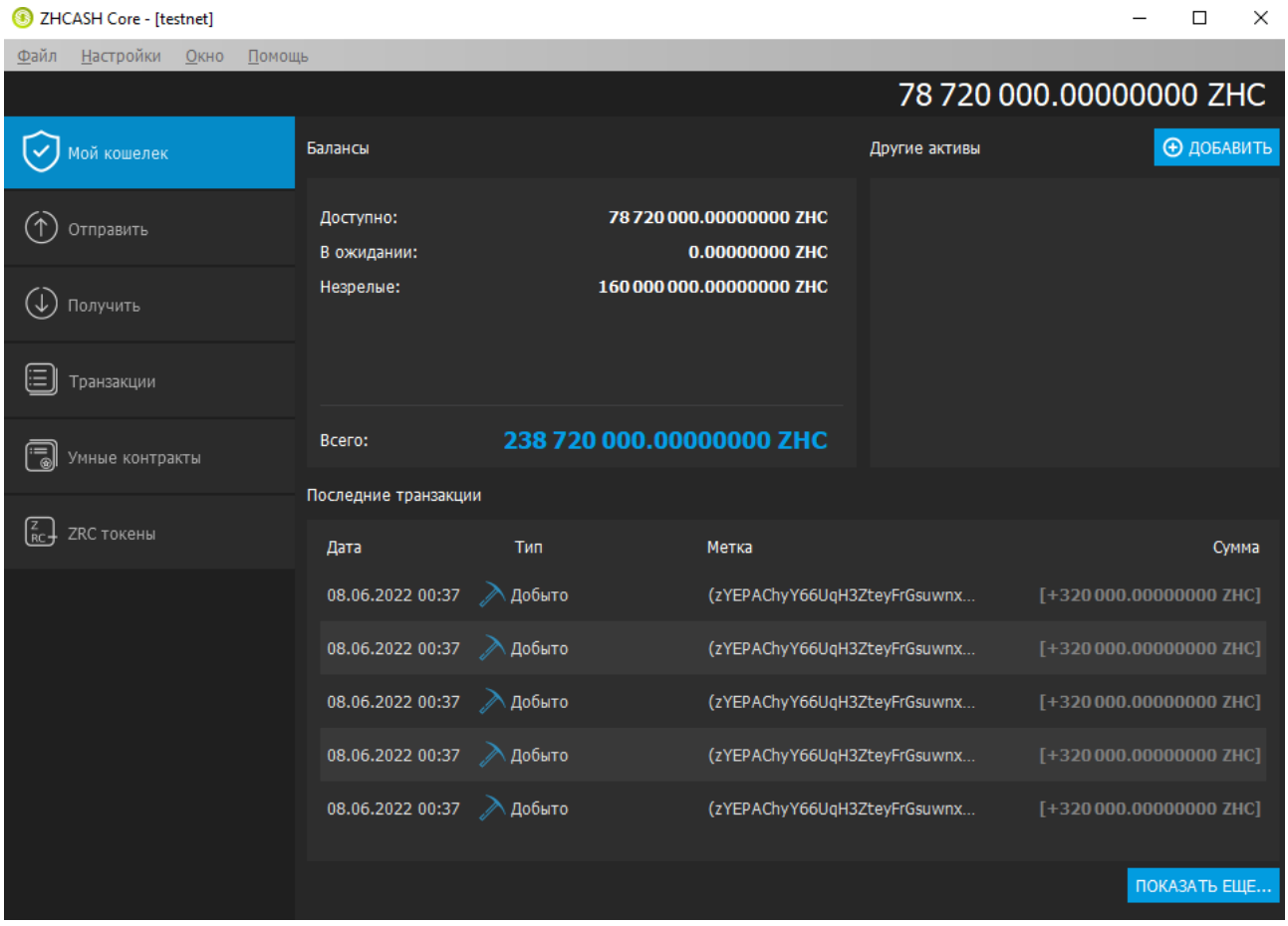


We go into the console, write generate 10 and get an error. In order to generate blocks, you need to run with the keys -testnet -deprecatedrpc=generate. We repeat again.

```
PS C:\Users\Yoga\Desktop> .\ZeroHour-Qt.exe -testnet
PS C:\Users\Yoga\Desktop> .\ZeroHour-Qt.exe -testnet -deprecatedrpc=generate
PS C:\Users\Yoga\Desktop>
```

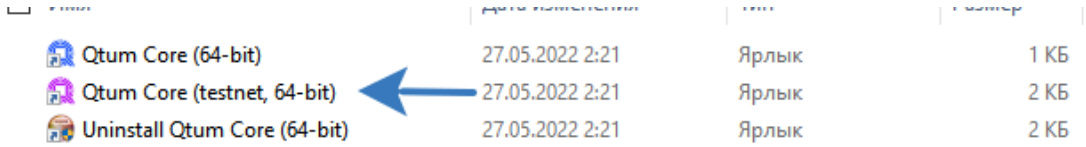
We poke this command in the console many, many times until something happens to the balance in the "Available" section. From immature shekels, the blockchain has indigestion.

After repeatedly raping the up key (to repeat the previous command) and enter, the blockchain is inflated and ready to go



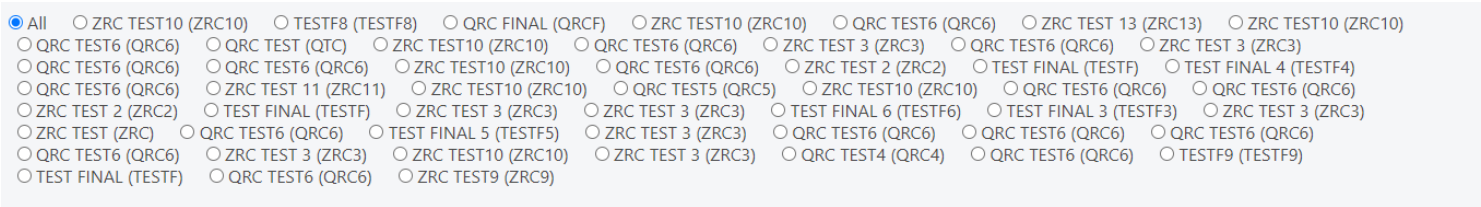
You can load smarts, check everything, then generate a block and test everything.

2) Using the QTUM testnet. They have a test network faucet where you can enter your address and you will get 50 ± 20 test coins. When downloading the wallet, a separate testnet wallet is immediately available.



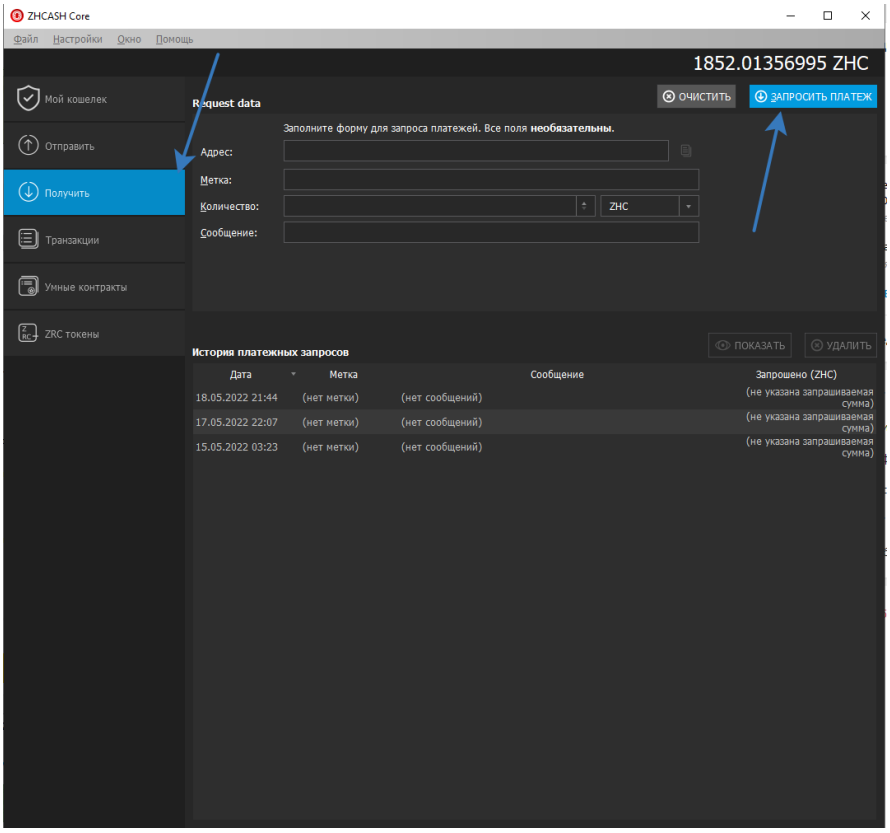
It takes about 2 hours to synchronize with the test blockchain. qtum testnet guidehttps://docs.qtum.site/en/Testnet-User-Guide.html

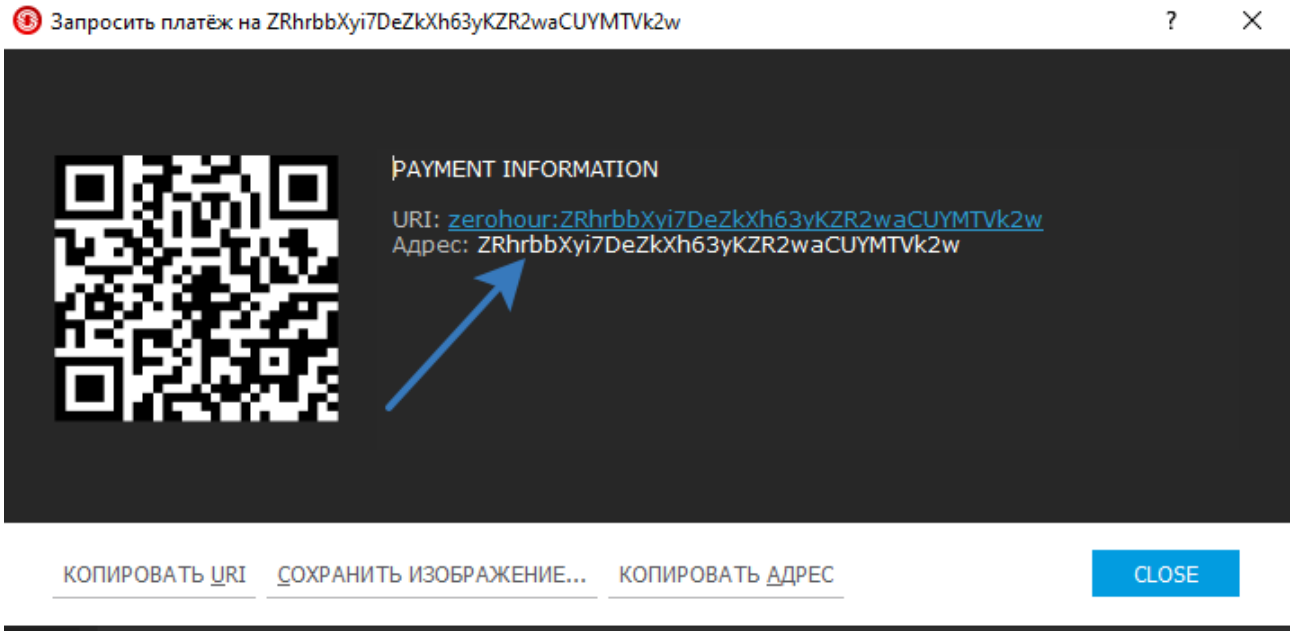
3) Creating dozens of test smarts in the main network (as the author did at the beginning. Therefore, he decided to write a guide), but this is deprecated. Below is the result of the third approach in testing.



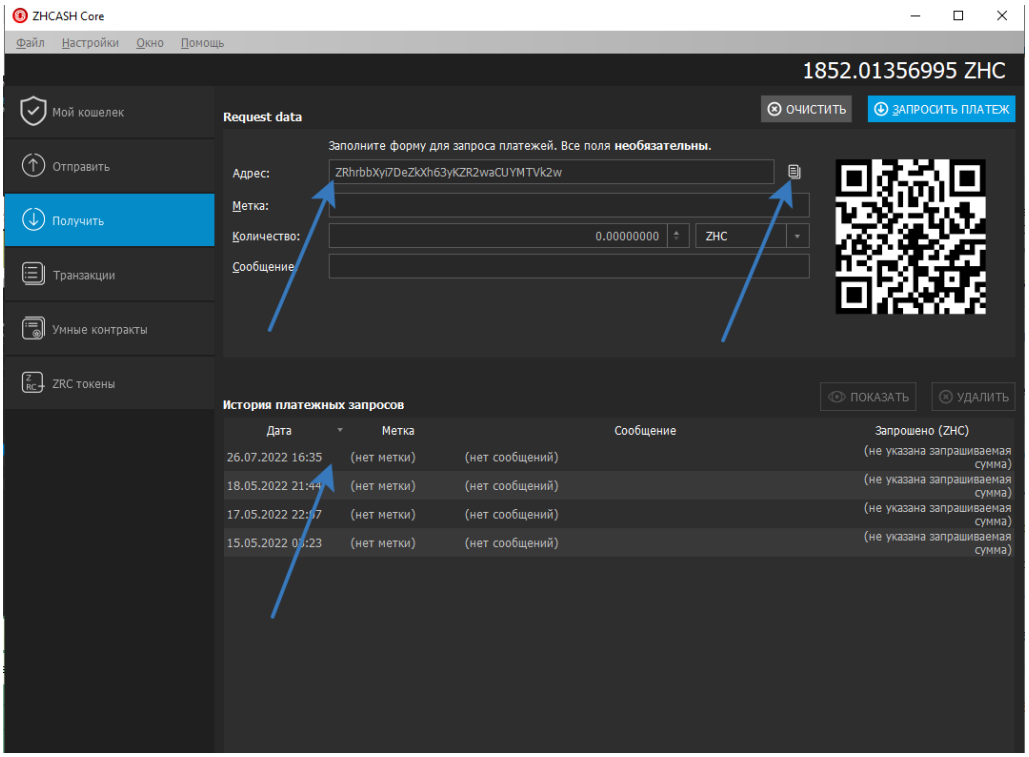
Getting a wallet address

To get your wallet number, you need to go to the "Get" tab and click the "REQUEST PAYMENT" button

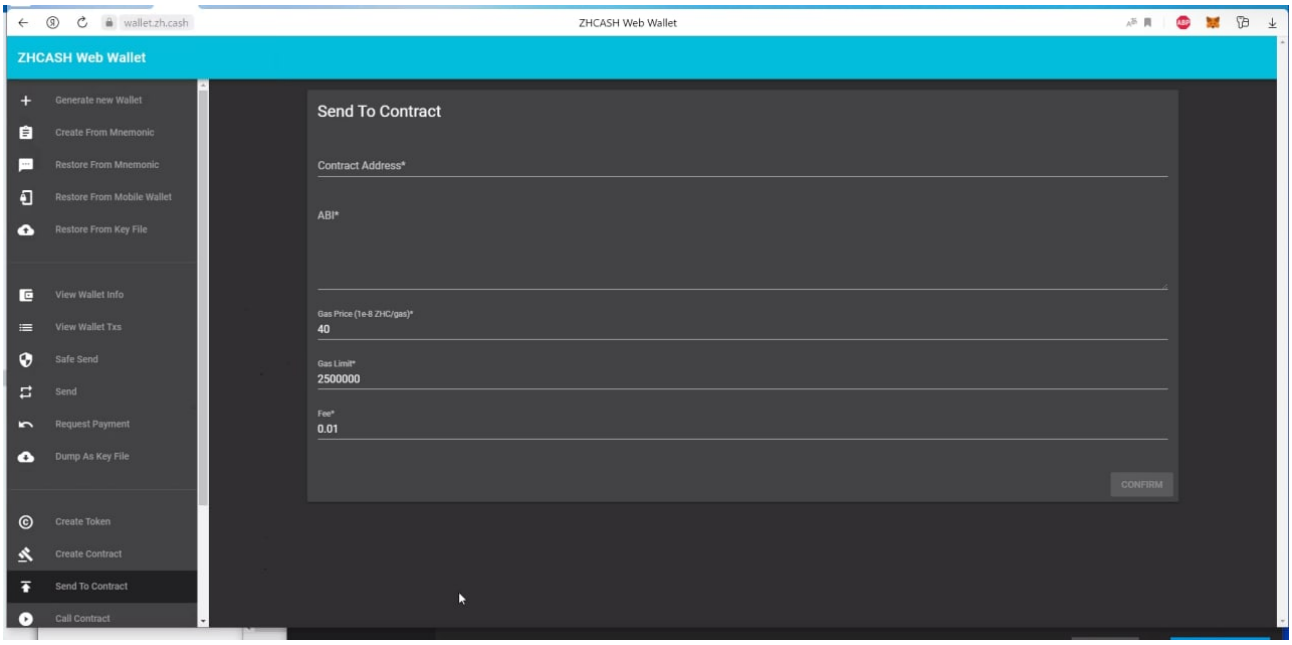




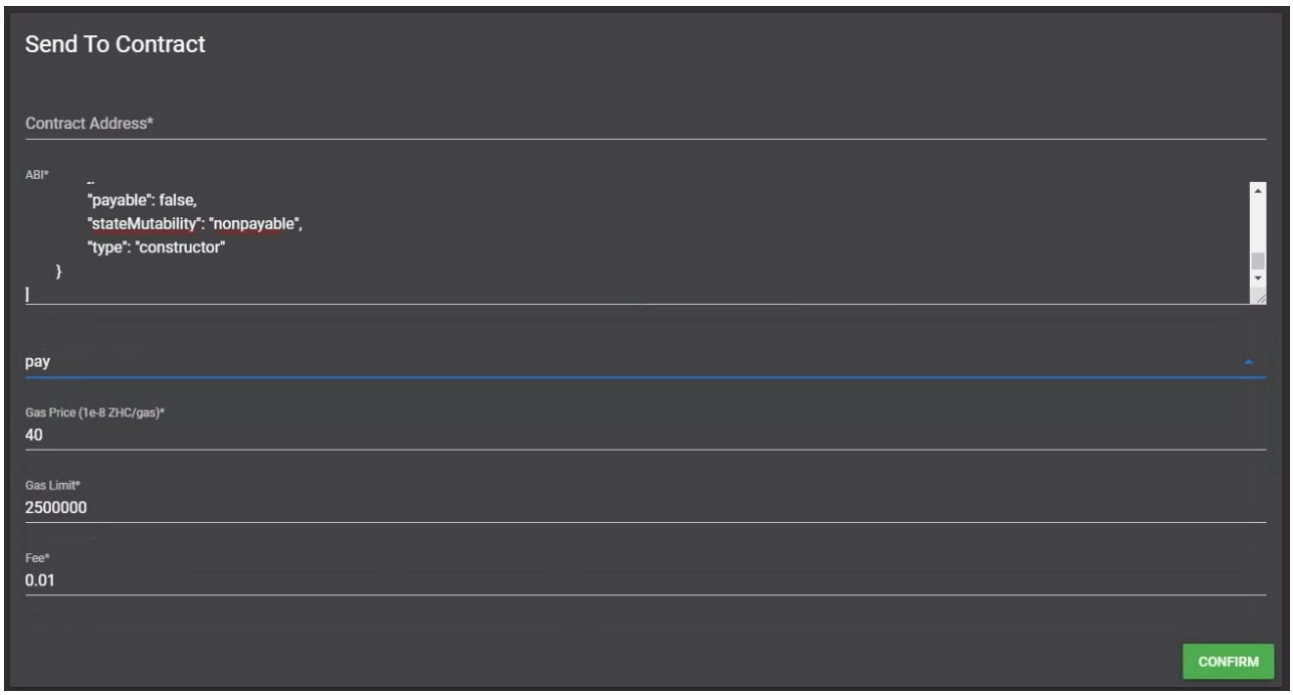
In order to copy the wallet number again, select the line in the history of payment requests and copy the Address that appears



WEB WALLET



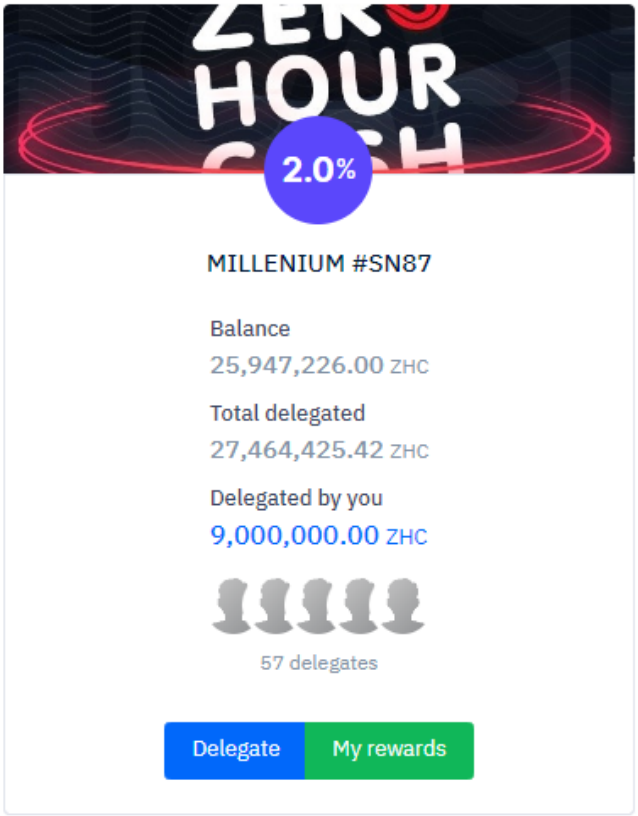
On the web version, you can't send zx with the sendtocontract command



And in general, you can't send zx to a smart contract. So far, such a function has not been added (as of 09/21/2022).

Otherwise, the web wallet has the same functionality as the node.

Large amounts of zx should be stored on the desktop QT version of the wallet, or delegated to pools and receive 2% per month. You can do this in the web console. For example, on node 87 **MILLENNIUM**https://zhcash.org/invests?mode=light&page=7



RPC errors:

error code: -4
error message:
Private key not available

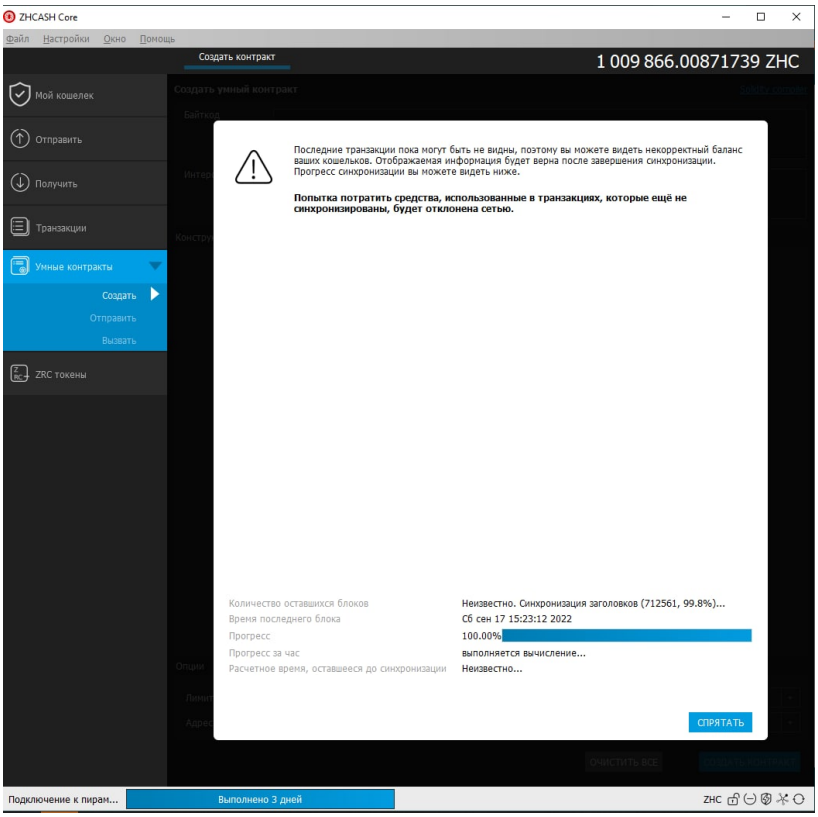
You did not specify your wallet number in the command

```
root@s3aa484ef:/usr/bin# ./zerohour-cli sendtocontract 1360e0bff6ba7cd703fa225a39aef92d5ba78340 21fcd3560000000000000000000
000000000000000000000000000000000000000000000400000000000000000000000000000000000000000000000000000000000000000000000000000c000000000000000
000000000000000000000000000000000000000000000000000300000000000000000000000000000000000000000000000000000000000000000000000000001000000000
000000000000000000000000000000000000000000000000000000000000000100000000000000000000000000000000000000000000000000000000000000000000010000
000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000184eb41e30b0d5974df3d1b2429fbdf728222a4
c000000000000000000000000184eb41e30b0d5974df3d1b2429fbdf728222a4c000000000000000000000000184eb41e30b0d5974df3d1b2429fbdf728
222a4c 0 200000 0.0000004 ZTwG2DpQNsRoczDJg4jXcZ43gCWoEXmmsc
error code: -4
error message:
Private key not available
```

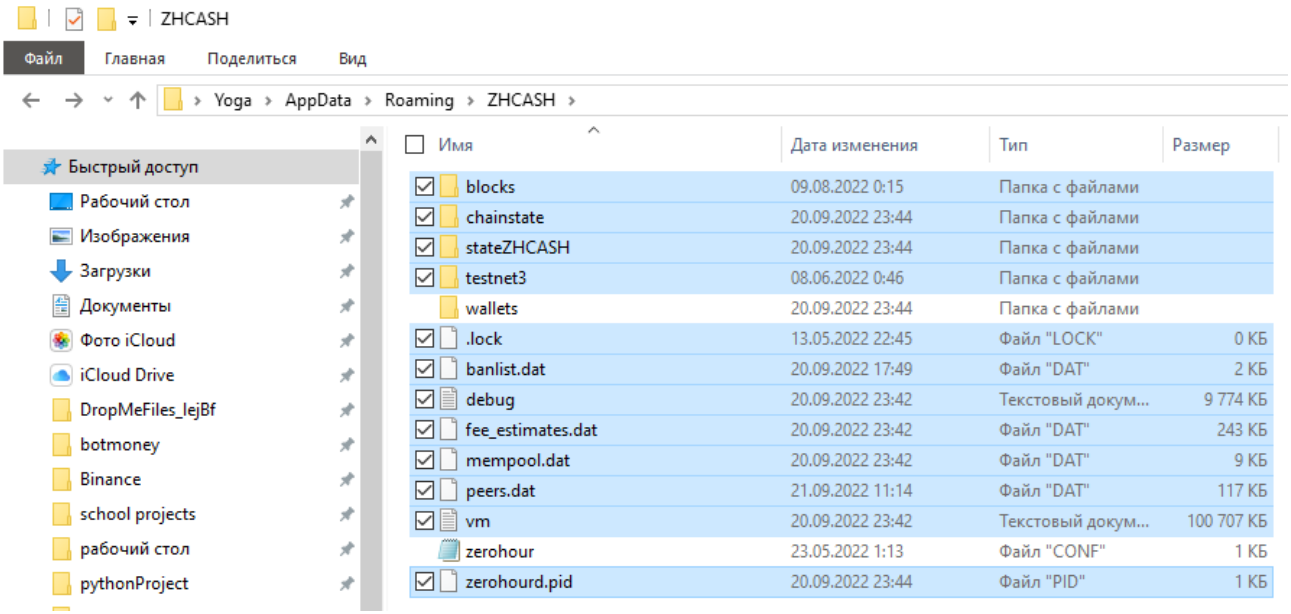
This is how it would be correct:

```
[1]
['184eb41e30b0d5974df3d1b2429fbdf728222a4c']
./bin/zerohour-cli sendtocontract 1360e0bff6ba7cd703fa225a39aef92d5ba78340 21fcd3560000000000000000000000000000000000000000
000000000000000000000004000000000000000000000000000000000000000000000000000000000000000000000000000800000000000000000000000000000000000000
000000000000000000000000000001000000000000000000000000000000000000000000000000000000000000000000000000100000000000000000000000000000000000
000000000000000000000000000000000100000000000000000000000184eb41e30b0d5974df3d1b2429fbdf728222a4c 0 100000 0.0000004 ZSAE6
X76tMhDVpMeGEQogWGzK2bmn9rNF3
0
```

If at some point the hard drive runs out of free space, the blockchain may stop updating.



To start a new synchronization, you need to delete all files from the ZHCASH folder (the same folder where the configuration file is located), except for the wallets folder and the zerohour configuration file



To do this, you need to close the qt wallet and run after the removal.

zeroscan API

Zeroscan has an api that can be used to extract any information from the blockchain. For example, all transactions for some wallet

https://ws.zeroscan.io/address/ZEFnGiHuwdSthnBA3cvAgPPFhhAKKqXQna/basic-txs

API description similar to Kutumovsky

https://github.com/qtumproject/qtuminfo-api

Installing the ZHCASH node on a server version of Ubuntu from version 18.04 and higher.

1. Download the archive with binaries and unpack it:

wget https://zh.cash/download/ZHCash-Console-Linux.zip && unzip ZHCash-Console-Linux.zip

Note: if the unzip archiver is not installed, install it with the command: apt install unzip (if the installation is under the root user) or with the command: sudo apt install unzip (if the installation is under another user with sudo rights).

2. Copy the daemon and client files to the user root:

cp ~/Console/zerohour-cli ~/ && cp ~/Console/zerohourd ~/

3. We give the rights to executable files for the user:

chmod u+x zerohourd && chmod u+x zerohour-cli (if installing as root) or: sudo chmod u+x zerohourd && sudo chmod u+x zerohour-cli (if installing as another user with sudo privileges).

4. Create a ZHCASH wallet data folder and a config file in it where we set the parameters for the daemon to work in the background and enable the staking mode in the wallet:

mkdir .zerohour && cd .zerohour && nano zerohour.conf

The nano text editor will start, we will write 2 parameters there:

daemon=1
staking=1
after that, save the configuration file with the CTRL + O keys, exit the nano editor CTRL + X

5. Go to the root of the user folder and launch the ZHCASH wallet:

cd && ./zerohourd

an inscription will appear that ZHCASH has started. Next, the wallet will start downloading blocks. See how many blocks have already been downloaded:

. /zerohour-cli getblockchaininfo | grep blocks

When the number of blocks equals the last block in the zhcash explorer, the wallet has successfully synced and is ready for staking.

Conclusion

The author wrote a smart for the LIFT token https://github.com/dimaystinov/Token-LIFT-ZHCASH

The author is grateful to the initiators of the creation of the LIFT token https://t.me/lift_club with addressf180d0a911d09853685764a9ad6d366398c50656 Nicholas, Arjun and Denis.

To the chief blockchain engineer Zx Roman, programmer Alex, developer Mike Gurov for answering stupid questions that formed the basis of this guide.

@QtumLeandro (From chathttps://t.me/qtumofficial) for the answer that you still need to send data to smart with the sendtocontract command.

Raul @kt2090 for the guide on installing a node on a server via ssh

Donations are accepted in ZHC shekels per wallet:

ZEFnGiHuwdSthnBA3cvAgPPFhhAKKqXQna