

## ZHCASH Beginner's Guide

"But I don't want to go among mad people," Alice remarked.  
"Oh, you can't help that," said the Cat: "we're all mad here. I'm mad. You're mad."

"How do you know I'm mad?" said Alice.  
"You must be," said the Cat, "or you wouldn't have come here."

—Lewis Carroll, Alice in Wonderland

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

Author

### DISCLAIMER

This version of the guide is unofficial and was written by the community in the art house style

### Definitions:

- 1) zh is the 5th generation ZHCASH blockchain. The result of the crossing (love) of bitcoin and ether, which has undergone genetic modifications <https://zh.cash/> . It is short for "ZHCASH blockchain"
- 2) The shekel is the main ZHC coin of the ZHCASH blockchain.
- 3) Drop shekels — transfer to my wallet ZHC
- 4) Drop off the LIFT – transfer the LIFT tokens to my wallet
- 5) ZRC20 is an analogue of the ERC20 and QRC20 (QTUM) standards
- 7) QTUM is the parent of zh <https://qtum.org/en> . Older by 2 years. 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 you will be quickly answered to any question.
- 8) Console – a command line in the zh wallet terminal or a program with a zerohour-cli command line interface. This is not a website <https://zhcash.org/> . It is used for entering commands and interacting with the blockchain. The API is on the site <https://zh.cash/docs/en/ZHCash-RPC-API/>
- 9) hex — the real address of the wallet in HEX format. It is according to him that tokens and shekels are accrued when interacting through a smart contract. It turns out when you enter the command

gethexaddress ZEFnGiHuwDStHnBA3cvAgPPFhhAKKqXQna

to the console. It will turn out

184eb41e30b0d5974df3d1b2429fbdf728222a4c

This is almost an ether address, except that there is no 0x in front of it. It is impossible to use smart in the code (it is not compiled), when adding 0x at the beginning it is not perceived in zh as a wallet. Use only this type of wallet (without 0x at the beginning), but do not use it directly in the smart code itself.

10) Remix – ether environment for smart development <https://remix.ethereum.org/>

12) Broken — smart (or transaction), which did not have enough gas and did not integrate into the blockchain. It is displayed in black in the explorer (zeroscan). For one broken, two unbroken are given or vice versa.

12) Explorer (zeroscan) - <https://zeroscan.io/>

13) Smart - smart contract for solidity

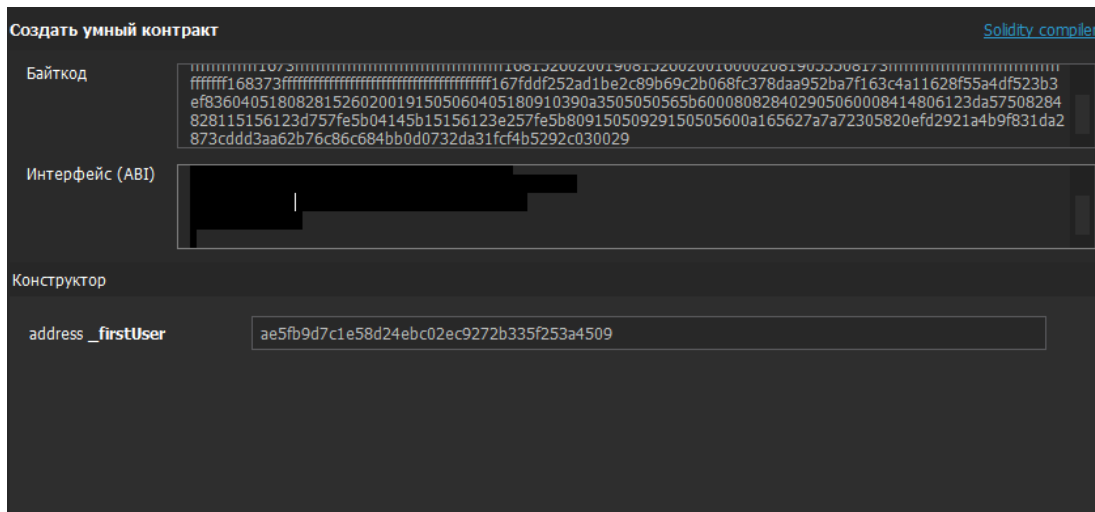
14) Nachill (to nachill)— get native coins from the test network, interest on tokens or receive a reward for mining a block. The main condition is that this happens in a relaxed way

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

### Smart development for zh

It is assumed that the reader already has an initial level of knowledge on solidity, which he can learn for example here <https://inaword.ru/smart-kontrakty/> or [https://www.tutorialspoint.com/solidity/solidity\\_variables.htm](https://www.tutorialspoint.com/solidity/solidity_variables.htm)

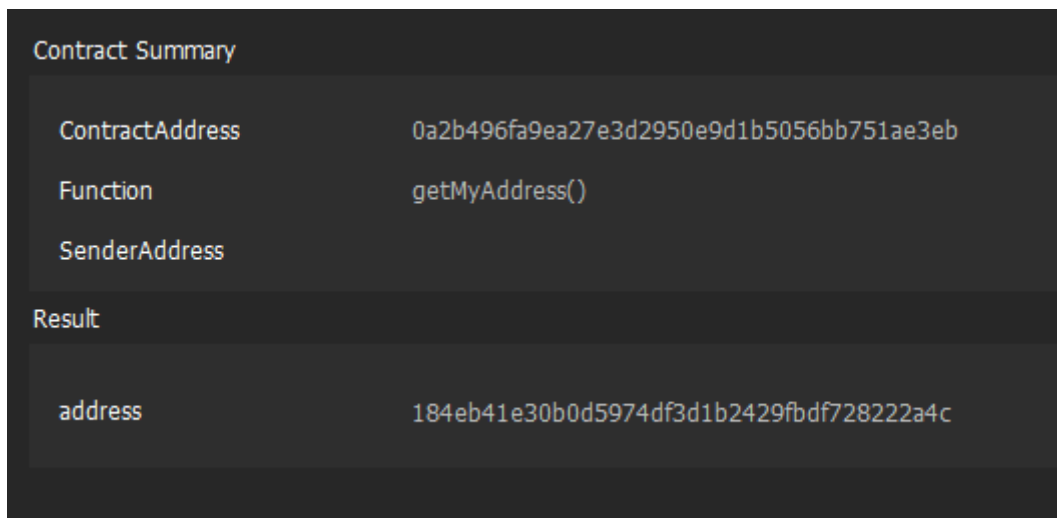
The zh system is identical to Ethereum, but there are some nuances. The address in zh is the same ether address, but without 0x. But this type of address cannot be specified directly in the remix code, so if we want to pass the value of the address (For example, the first user to be credited with a million tokens), then we can do this when creating a contract. At the same time, the address should be written in hex format



You can get hex through the console by command **gethexaddress** (take a closer look at the first line in the screenshot below)

```
15:23:20 [down arrow] gethexaddress ZEFnGiHuwdSthnBA3cvAgPPFhhAKKqXQna
15:23:20 [up arrow] 184eb41e30b0d5974df3d1b2429fbdf728222a4c
```

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



You can overtake hex in the classic form with the **fromhexaddress** command (look closely at the first line in the screenshot below)

```
21:00:24 [down arrow] fromhexaddress 184eb41e30b0d5974df3d1b2429fbdf728222a4c
21:00:24 [up arrow] ZEFnGiHuwdSthnBA3cvAgPPFhhAKKqXQna
```

With airdrop, 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 unloading it, there is no "extracted" icon

in transactions and it will be displayed in black on the zeroscan in the smart block), then you should increase the gas. Normal situations are shown below.

04.06.2022 22:16	Добыто	(ZbFkrDeGDr6EW1UyXBCGV7vSojEgpLF8fm)	[0.26905840]
04.06.2022 22:13	Отправка контракта	(fea82863035a4a5edf5fe8434a39b17e5ab22a05)	-1.44250400

469e8e0ff1aa688d3de869572cbaca3d2727405e5883877f74cddbfbb4453c1	3 confirmations	2022-06-04 22:17:04
ZbFkrDeGDr6EW1UyXBCGV7vSojEgpLF8fm	495.45034960 ZHC	Contract Create
Gas Back	→ eccecb4245cd6ddb4fb1bc4e24a1b8dea1c30e65	494.00784560 ZHC
Mint Tokens	→ ZJWpducUCKwX86yktUw6FjP61Sj9oLvQQ	
Mint Tokens	→ ZbFkrDeGDr6EW1UyXBCGV7vSojEgpLF8fm	0.26905840 ZHC
	→ ZEFnGiHuwD5thnBA3cvAgPPFhhAKKqXQna	999000000 TESTF9
	→ ZTwG2pQNsRocZJg4jXcZ43gCWoEXmmsc	1000000 TESTF9
		Fee 1.1734456 ZHC

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 50,000 gas, but it passed with 100,000 gas. The recommended gas for any transactions is in zh 250000, in qtum 100000.

[vm] from: 0xab8...35cb2 to: ZRC20Token.addNewUser(address,uint256) 0xd91...39138 value: 0 wei data: 0x10f...0000a logs: 1 hash: 0xa8f...9137e	Debug
status	true Transaction mined and execution succeed
transaction hash	0xaaff451b9a1bcbca76d53ceeb349f28fdd6d3190667dbf92ab220de3137e
from	0xab8443f4d9c6d1c0f9b449a677dd0315035cb2
to	ZRC20Token.addNewUser(address,uint256) 0xd91450CE52036E254917e481e84e9943F9138
gas	42005 gas ← МИНИМАЛЬНО НЕОБХОДИМЫЙ ГАЗ
transaction cost	36526 gas
execution cost	36526 gas
input	0x10f...0000a ← КОМАНДА ДЛЯ КОМАНДНОЙ СТРОКИ
decoded input	{ "address": "0x4B209928c481177ec7E8F571c0cE3A9e22C02db", "uint256_value": "10" }

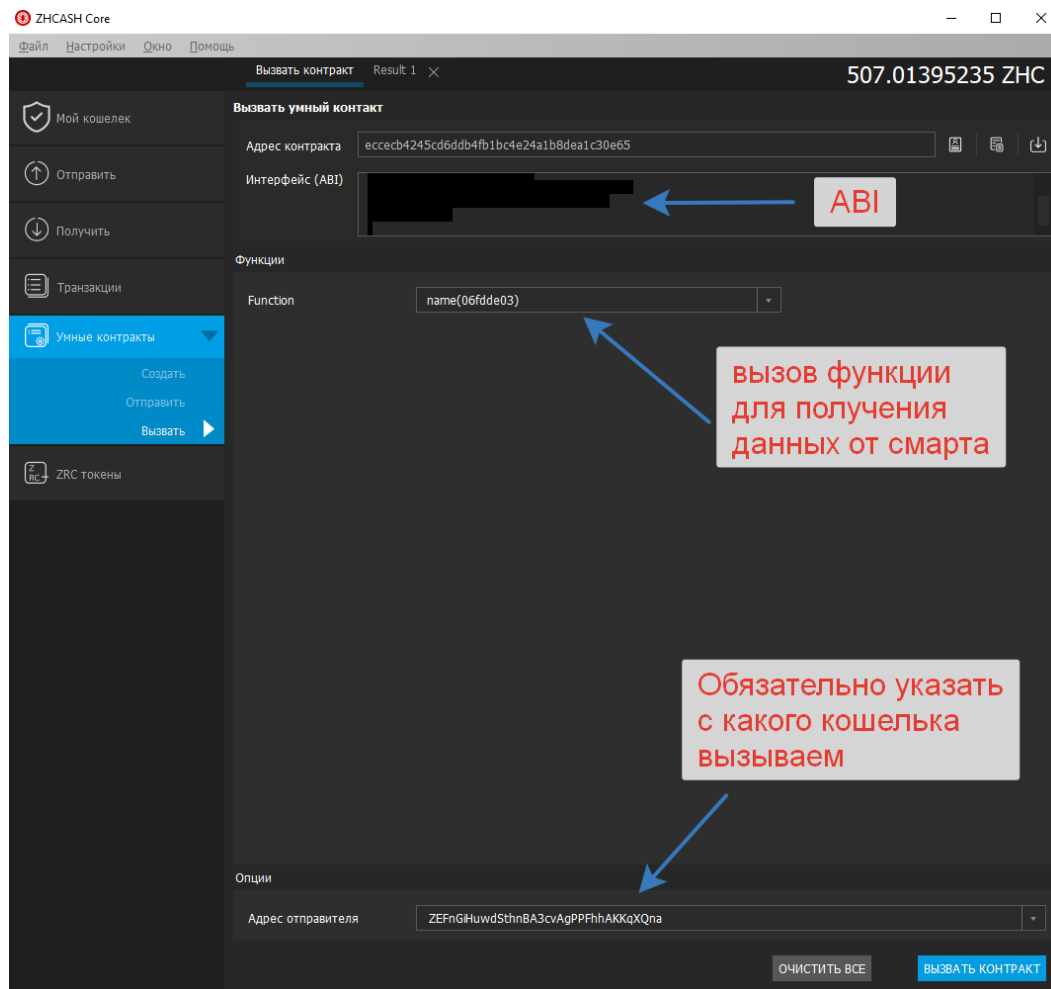
Also there you can find out which command (command code) is being executed at the same time. This is necessary to interact with the blockchain through the console.

Zh supports the latest version of solidity, but it is recommended to use the 0.7 version. 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 gas value. And when compiling on versions 0.7, even the standard example of smart beats if you leave the gas 250,000.

It is recommended to download smart cards and send data (sendtocontract) to smart from a wallet with a balance of no more than 100 shekels, because there is a high probability of the entire balance being fucked up. The author has repeatedly encountered this. This rule does not apply to calling contract functions (callcontract) and single token transfers 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 begins (from 1 thousand to 400 thousand). So you can drain the entire balance of the node (1 million or more easily) on the drop by sending tokens to

60 users in one block. As a result, the author made a drop newsletter to 20 users in one block (with a delay of 10 minutes). 200 users are sent out per hour, which is acceptable. In 4 hours, the drop started for everyone. There are still ways to circumvent this restriction and start tokens for thousands of addresses instantly, but we will leave this exercise to the reader.

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

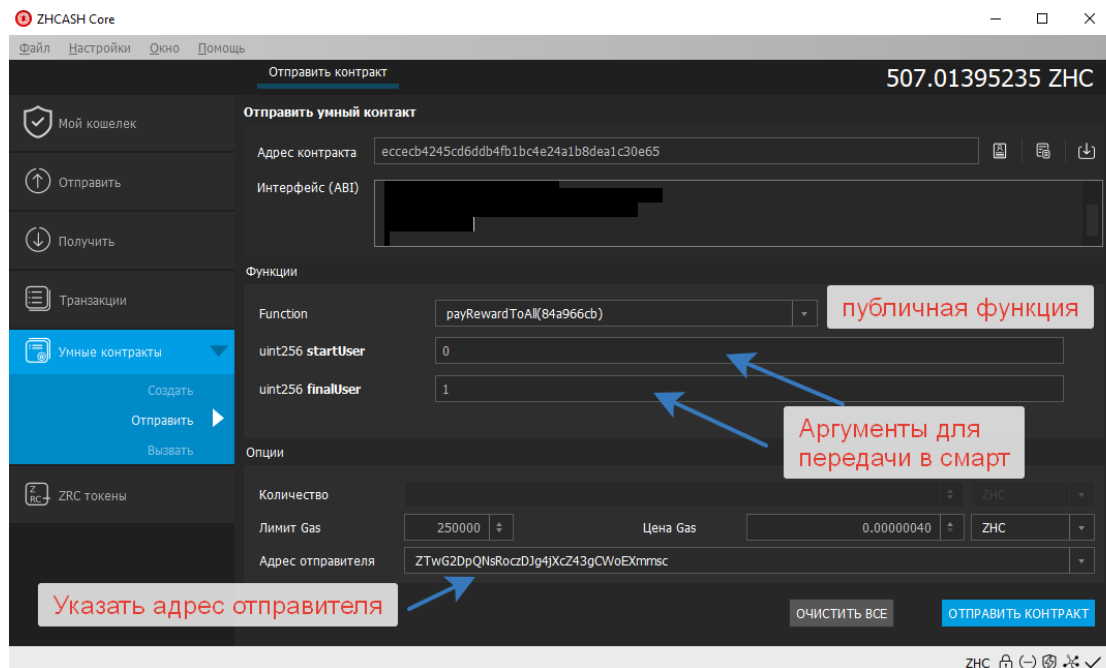


If we want to get the data, we should use the "Call" tab. We get the following result.

Contract Summary	
ContractAddress	ecceb4245cd6ddb4fb1bc4e24a1b8dea1c30e65
Function	name()
SenderAddress	
Result	
string	TESTF9

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 "Send" tab



If we want to pay a reward to all users, then we should also pay in portions of 20 transactions per block. And put a greater gas value. After such an appeal, we get the following.

Contract Summary	
Transaction ID	9972da26f233d7e963aa1ca93eef7bb341c2a89571c2762545d05c92b8f698f8
SenderAddress	ZTwG2DpQNsRoczDJg4jXcZ43gCWoEXmmmc
Hash160	ae5fb9d7c1e58d24ebc02ec9272b335f253a4509

After waiting for a new block for two minutes and going into "Transactions", we will notice that the "Extracted" icon will appear

05.06.2022 16:03	Добыто	(ZL25qnzUA7uz7kQBYC3vmZAt8jgS19UXwL)	[0.07348320]
05.06.2022 16:02	Отправка контракта	(57884d4449512ede1e192415c8cf4029969fd5f0)	-0.10188400

This means that the data has been sent successfully.

Now let's look at the interaction with the blockchain through the console.

To do this, there are two commands: call contract to receive data and send to contract to send data to smart. Below is an example of usage.

```
callcontract  eccecb4245cd6ddb4fb1bc4e24a1b8dea1c30e65
06fdde03
```

Debug

```
call [call] from: 0x5B38Da6a701c568545dCfcB03FcB875f56beddC4 to: ZRC20Token.name() data: 0x06f...dde03

from 0x5B38Da6a701c568545dCfcB03FcB875f56beddC4

to ZRC20Token.name() 0xd91450CE52D286f254917e481e844e9942F29128

execution cost 21682 gas (Cost only applies when called by a contract)

input 0x06f...dde03 ← команда

decoded input {}

decoded output {
  "0": "string: TESTF10"
}

logs []
```

```
callcontract eccecb4245cd6ddb4fblbc4e24alb8dealc30e65 0x06fddde03
{
  "address": "eccecb4245cd6ddb4fblbc4e24alb8dealc30e65",
  "executionResult": {
    "gasUsed": 21046,
    "excepted": "Revert",
    "newAddress": "eccecb4245cd6ddb4fblbc4e24alb8dealc30e65",
    "output": "",
```

[illegible]

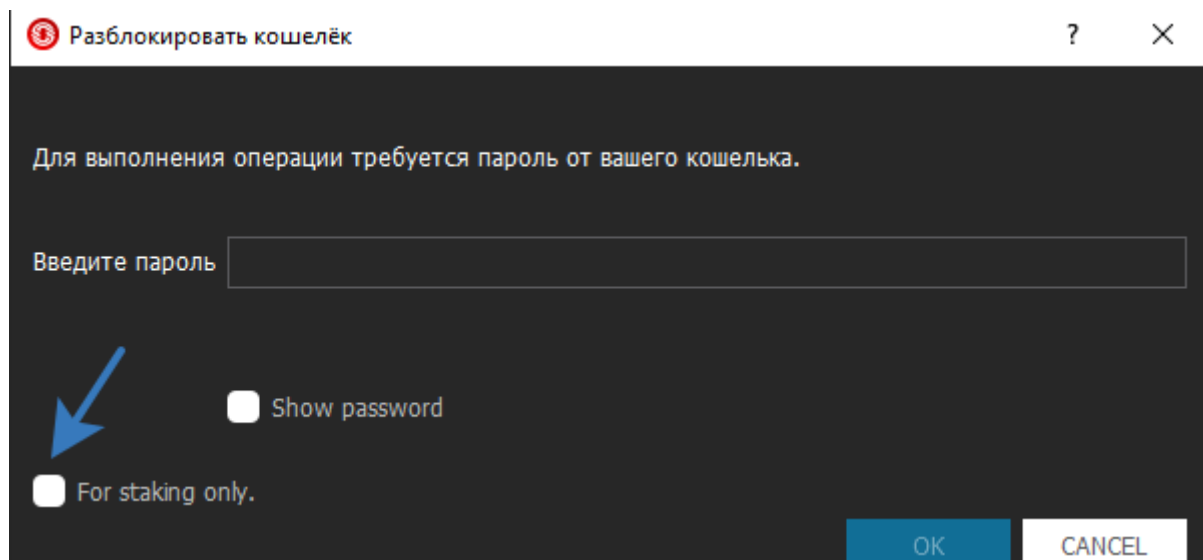
We will get some data, which can then be distilled into a string. Below is a screenshot for send to contract

[illegible]

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

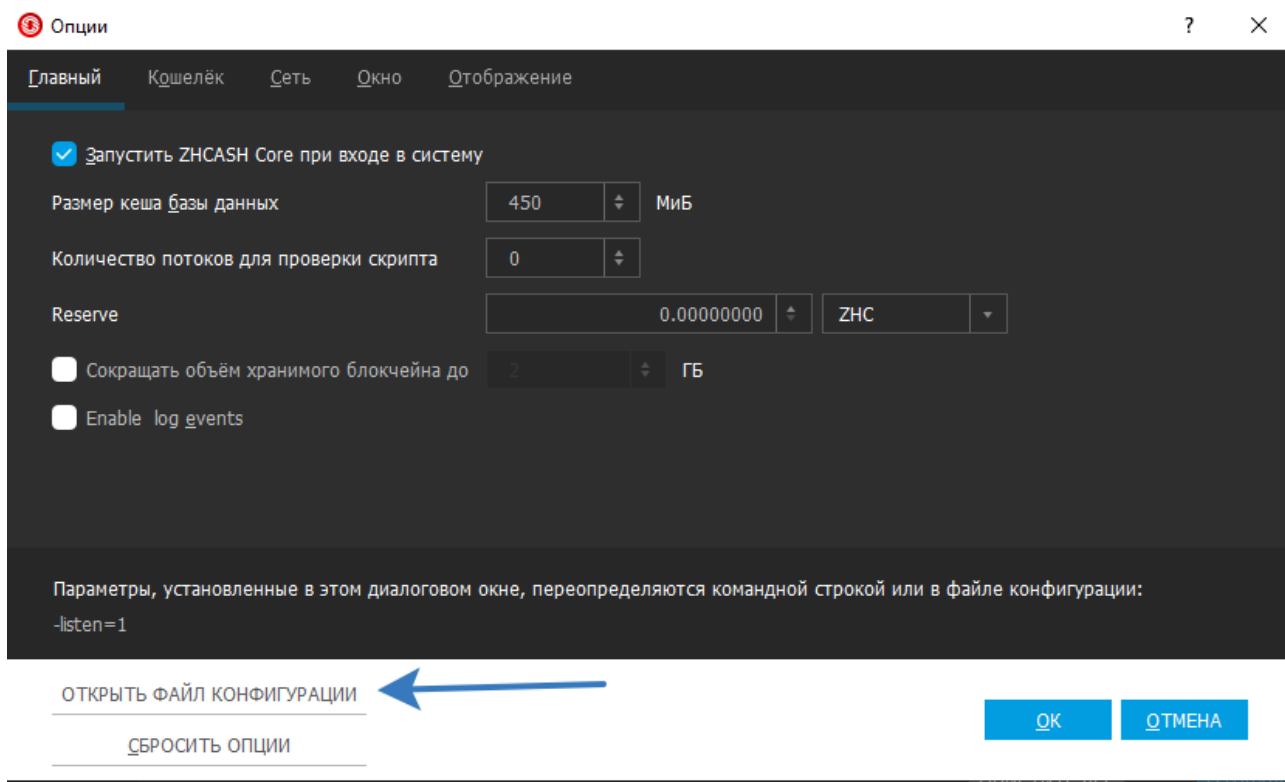


Then unlock the wallet and uncheck “For staking only”



### Add a configuration file in the parameters





And save the following text, where in the last argument of **rpc password** to 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
#rpccallowip=17.2.7.12
#rpccallowip=17.2.7.11
rpccallowip=127.0.0.1
rpcuser=zerohour-rpcuser
rpcpassword=
```

```
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
#rpccallowip=17.2.7.12
#rpccallowip=17.2.7.11
rpccallowip=127.0.0.1
rpcuser=zerohour-rpcuser
rpcpassword=1123581321
```

[illegible]

Three approaches can be used to test smart.

Windows PowerShell

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

ZHCASH Core - [testnet]

Файл Настройки Окно Помощь

0.00000000 ZHC

Мой кошелек

Отправить

Получить

Транзакции

Умные контракты

ZRC токены

Баланс

Доступно: 0.00000000 ZHC

В ожидании: 0.00000000 ZHC

Незрелые: 3 200 000.00000000 ZHC

Всего: 3 200 000.00000000 ZHC

Другие активы

ДОБАВИТЬ

Последние транзакции

Дата	Тип	Метка	Сумма
28.05.2022 23:56	Добыто	(z)brZjNFnN3DHEPHVN9Uk6f6pM8...	[+320 000.00000000 ZHC]
28.05.2022 23:56	Добыто	(z)brZjNFnN3DHEPHVN9Uk6f6pM8...	[+320 000.00000000 ZHC]
28.05.2022 23:56	Добыто	(z)brZjNFnN3DHEPHVN9Uk6f6pM8...	[+320 000.00000000 ZHC]
28.05.2022 23:56	Добыто	(z)brZjNFnN3DHEPHVN9Uk6f6pM8...	[+320 000.00000000 ZHC]
28.05.2022 23:53	Добыто	(z)wSURtGRLdDDemTFd3orJE7zBz9...	[+320 000.00000000 ZHC]

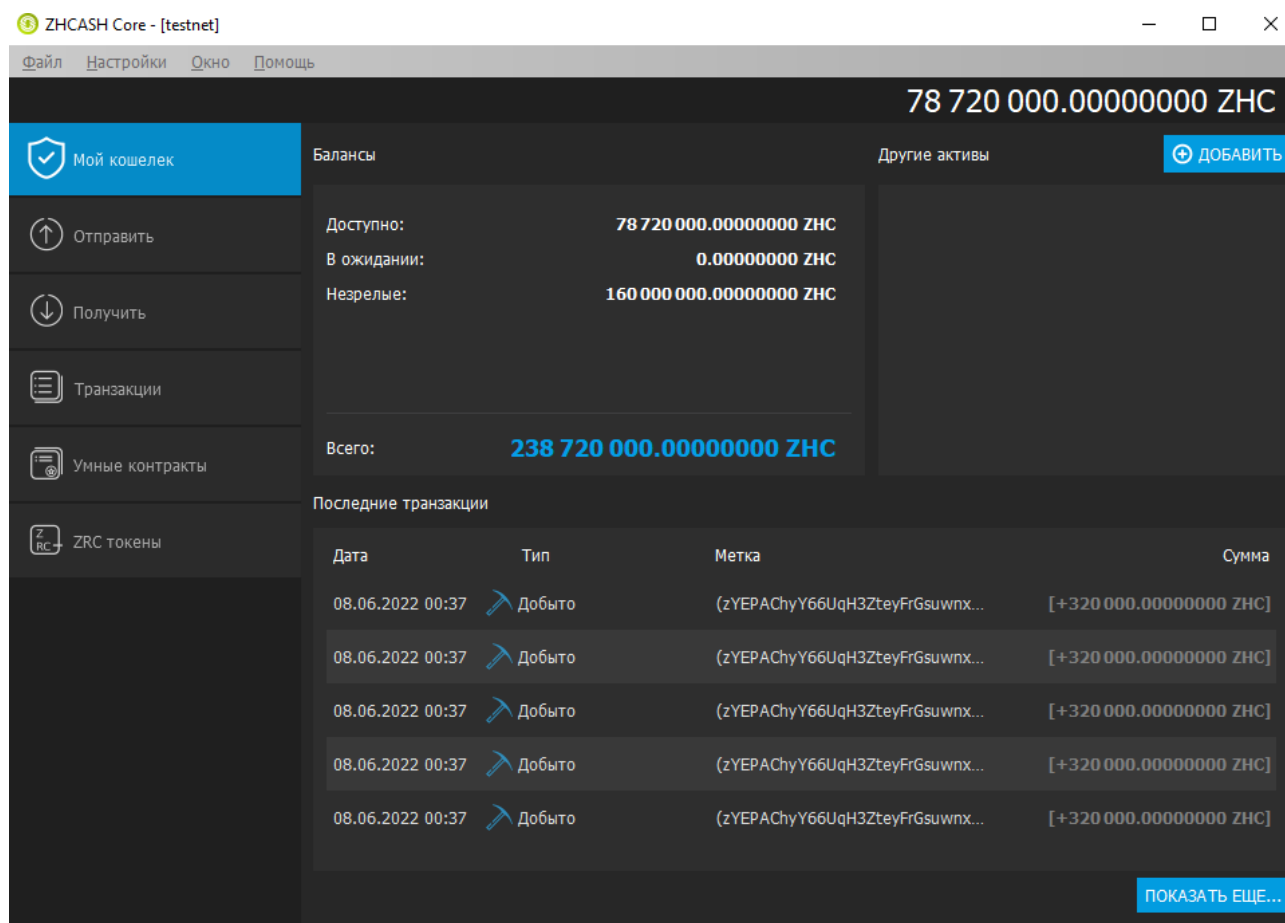
We go to the console, write **generate 10** and get an error. In order to generate blocks, you need to run with the keys **-testnet -deprecatedrpc=generate** We repeat it again.

```
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 last command) and enter, the blockchain is ready to work



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

2) Using the QTUM test network. They have a tap for the test network, where you can specify your address and you will be credited with 50 = 20 test coins. When downloading a wallet, a separate test net wallet is immediately available.

 Qtum Core (64-bit)	27.05.2022 2:21	Ярлык	1 КБ
 Qtum Core (testnet, 64-bit) 	27.05.2022 2:21	Ярлык	2 КБ
 Uninstall Qtum Core (64-bit)	27.05.2022 2:21	Ярлык	2 КБ

It takes about 2 hours to synchronize with the testing blockchain. Guide to the test there is no qtum <https://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 condemned. Below is the result of the third approach in testing.

☒ All  
 ☐ ZRC TEST10 (ZRC10)  
 ☐ TESTF8 (TESTF8)  
 ☐ QRC FINAL (QRCF)  
 ☐ ZRC TEST10 (ZRC10)  
 ☐ QRC TEST6 (QRC6)  
 ☐ ZRC TEST 13 (ZRC13)  
 ☐ ZRC TEST10 (ZRC10)  
 ☐ QRC TEST6 (QRC6)  
 ☐ QRC TEST (QTC)  
 ☐ ZRC TEST10 (ZRC10)  
 ☐ QRC TEST6 (QRC6)  
 ☐ ZRC TEST 3 (ZRC3)  
 ☐ QRC TEST6 (QRC6)  
 ☐ ZRC TEST 3 (ZRC3)  
 ☐ QRC TEST6 (QRC6)  
 ☐ ZRC TEST 11 (ZRC11)  
 ☐ ZRC TEST10 (ZRC10)  
 ☐ QRC TEST5 (QRC5)  
 ☐ ZRC TEST10 (ZRC10)  
 ☐ QRC TEST6 (QRC6)  
 ☐ QRC TEST6 (QRC6)  
 ☐ ZRC TEST 2 (ZRC2)  
 ☐ TEST FINAL (TESTF)  
 ☐ TEST FINAL 4 (TESTF4)  
 ☐ ZRC TEST 2 (ZRC2)  
 ☐ TEST FINAL (TESTF)  
 ☐ ZRC TEST 3 (ZRC3)  
 ☐ ZRC TEST 3 (ZRC3)  
 ☐ TEST FINAL 6 (TESTF6)  
 ☐ TEST FINAL 3 (TESTF3)  
 ☐ ZRC TEST 3 (ZRC3)  
 ☐ ZRC TEST (ZRC)  
 ☐ QRC TEST6 (QRC6)  
 ☐ TEST FINAL 5 (TESTF5)  
 ☐ ZRC TEST 3 (ZRC3)  
 ☐ QRC TEST6 (QRC6)  
 ☐ QRC TEST6 (QRC6)  
 ☐ QRC TEST6 (QRC6)  
 ☐ QRC TEST6 (QRC6)  
 ☐ ZRC TEST 3 (ZRC3)  
 ☐ ZRC TEST10 (ZRC10)  
 ☐ ZRC TEST 3 (ZRC3)  
 ☐ QRC TEST4 (QRC4)  
 ☐ QRC TEST6 (QRC6)  
 ☐ TESTF9 (TESTF9)  
 ☐ TEST FINAL (TESTF)  
 ☐ QRC TEST6 (QRC6)  
 ☐ ZRC TEST9 (ZRC9)

## Conclusion

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

The author expresses gratitude to the initiators of the creation of the LIFT token [https://t.me/lift\\_club](https://t.me/lift_club) with the address

f180d0a911d09853685764a9ad6d366398c50656

This smart was used in examples

To Nikolai, Arjun and Denis.

To the chief engineer of the blockchain zh Roman, programmer Alex for the answers to the stupid questions that formed the basis of this guide.

Alexey, the President of the zero gravity Foundation, for the request to translate the guide into English, so that you can read it

@QtumLeandro (From the chat <https://t.me/qtumofficial> ) for the answer that it is still necessary to send data to smart with the sendtocontract command.

Donations are accepted in shekels ZHC per purse:

ZEFnGiHuwDStHnBA3cvAgPPFhhAKKqXQna