

## Гайд новичка ZHCASH

План, что и говорить, был превосходный:  
простой и ясный, лучше не придумать.  
Недостаток у него был только один:  
было совершенно неизвестно,  
как привести его в исполнение.

Алиса в стране чудес

Если бы у меня был этот гайд,  
то я бы закончил смарты на месяцы раньше

Автор

### ДИСКЛЕЙМЕР

Данная версия гайда неофициальна и выполнена в стиле артхаус  
Редакция 1.4 на 02.04.2023

В данном гайде описаны нюансы по написанию смартов под zh, возможные ошибки начинающих разработчиков, как работать с нодой через терминал и питон, запуск своего локального тестового блокчейна, нюансы веб кошелька, возможные ошибки RPC и установка ноды по ssh на сервере убунты

Определения:

- 1) zh – блокчейн 5го поколения ZHCASH. Результат скрещивания (любви) биткоина и эфира, прошедший генные модификации <https://zh.cash/> . Произносится как «зх». Является противоположностью «хз».
- 2) зх — основная монета ZHC блокчейна ZHCASH.
- 3) Зерошка — аналог сатоши в биткоине. Равна  $10^{-8}$  зх
- 4) лифт – токен проекта LIFT
- 5) ZRC20 – аналог стандартов ERC20 и QRC20 (QTUM)
- 7) QTUM – родитель zh <https://qtum.org/en>. Старше на 2 года. Был взят самый перспективный блокчейн в мире (у китайцев) на момент 2019 года и существенно улучшен до лучшего в мире zh. Есть хорошая поддержка в телеграмме, где вам быстро ответят на любой вопрос.
- 8) Консоль – командная строка в терминале кошелька zh или программа с интерфейсом в командной строке zerohour-cli. Это не сайт <https://zhcash.org/>.

Используется для ввода команд и взаимодействия с блокчейном. API есть на сайте <https://zh.cash/docs/en/ZHCash-RPC-API/>

8\*) Веб-консоль - сайт <https://zhcash.org/>.

9) hex — реальный адрес кошелька в HEX формате. Именно по нему происходит начисление токенов и шекелей при взаимодействии через смарт контракт. Получается при введении команды  
`gethexaddress ZEFnGiHuwdSthnBA3cvAgPPFhhAKKqXQna`  
в консоль. Получится `184eb41e30b0d5974df3d1b2429fbdf728222a4c`  
Это почти эфировский адрес, за исключением того, что перед ним не стоит 0x. Использовать в коде смарта нельзя (не компилируется), при добавлении 0x в начале не воспринимается в zh как кошелек. Пользоваться исключительно таким видом кошелька (без 0x в начале), но не применять непосредственно в самом коде смарта.

10) Ремикс – эфировая среда для разработки смарта <https://remix.ethereum.org/>

12) Битый — смарт (или транзакция), которому не хватило газа и он не вошел в блокчейн. Отображается черным цветом в эксплорере (зероскан). За одного битого двух небитых дают или же наоборот.

12) Эксплорер (зероскан) — <https://zeroscan.io> его api описано в конце

13) Смарт — смартконтракт на solidity

14) Начилить (начил)— получить нативные монеты из тестовой сети, проценты по токенам или получение вознаграждения за майнинг блока. Главное условие чтобы это происходило на расслабоне

15) Дроп — централизованная раздача токенов или шекелей в начале запуска смарта. Является сокращением от air drop.

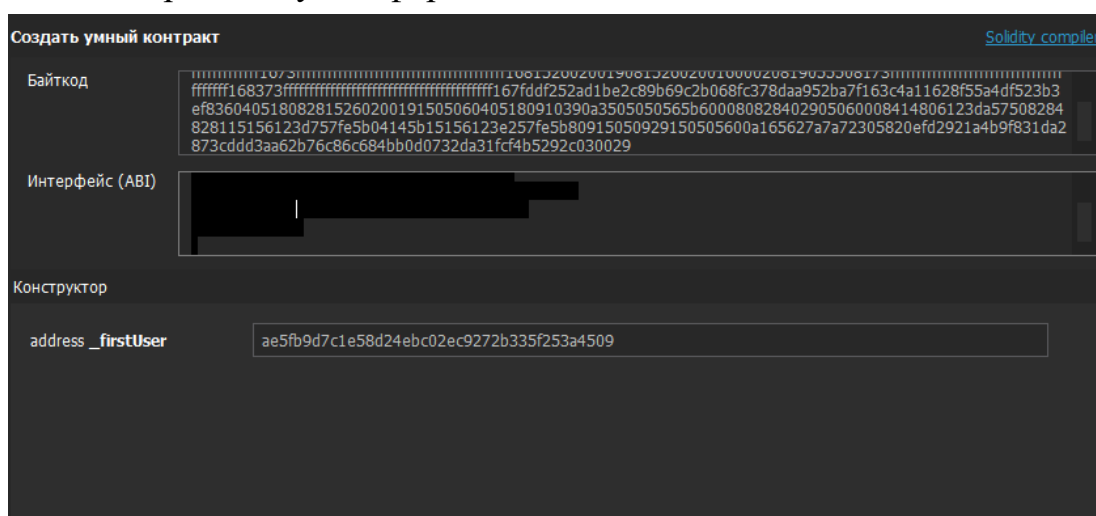
16) UTXO (Unspent Transaction Output) - выход неизрасходованных транзакций <https://2bitcoins.ru/chto-takoe-utxo-i-zachem-on-nuzhen/>

При отправке 1 зх с баланса 10 зх на адрес пересылается вся сумма, а сдача 9 зх возвращается в следующем блоке. Не даёт сделать много транзакций в блокчейне, но и повышает безопасность блокчейна. Используется в биткоине. На эфире используется Система учетных записей. Сравнение этих систем описано здесь <https://russianblogs.com/article/24511021659/>

## Разработка смарт под zh

Предполагается, что читатель уже имеет начальный уровень знаний по solidity, который он может почерпнуть например здесь <https://inaword.ru/smart-kontrakty/> или [https://www.tutorialspoint.com/solidity/solidity\\_variables.htm](https://www.tutorialspoint.com/solidity/solidity_variables.htm)

Система смартов zh идентична Ethereum, но есть некоторые нюансы. Адрес в zh это тот же эфировский адрес, но без 0x. Но такой тип адреса нельзя указать непосредственно в коде ремикса, поэтому если мы хотим передать значение адреса (Например, первого пользователю, которому будет начислен миллион токенов), то делать мы это можем при создании контракта. При этом прописывать адрес следует в формате hex



Создать умный контракт Solidity compiler

Байткод

Интерфейс (ABI)

Конструктор

address \_firstUser ae5fb9d7c1e58d24ebc02ec9272b335f253a4509

Получить hex можно через консоль gethexaddress (приглядывайтесь к первой строчке на скрине ниже)

```
15:23:20 gethexaddress ZEFnGiHuwDStHnBA3cvAgPPFhhAKKqXQna
15:23:20 184eb41e30b0d5974df3d1b2429fbdf728222a4c
```

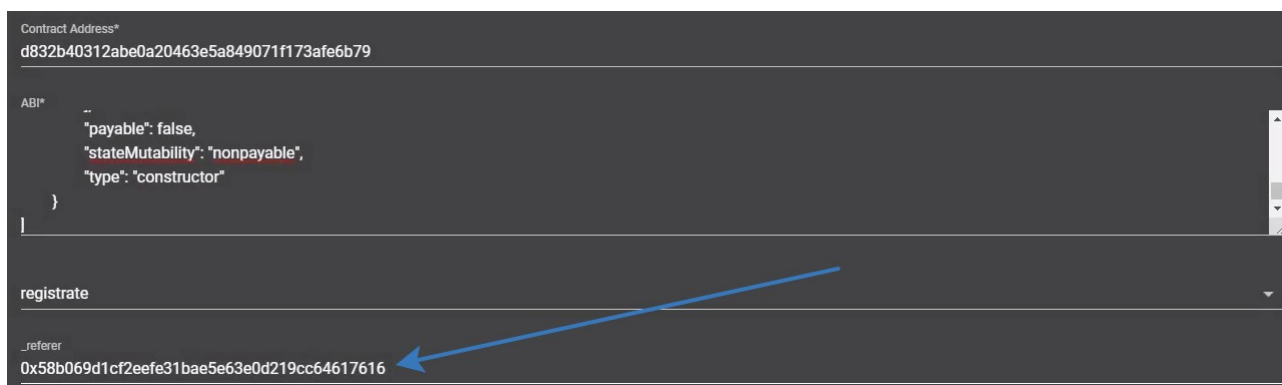
Воспользовавшись функцией моего смарт получения своего адреса кошелька, заметим, что он будет также в hex формате

Contract Summary	
ContractAddress	0a2b496fa9ea27e3d2950e9d1b5056bb751ae3eb
Function	getMyAddress()
SenderAddress	
Result	
address	184eb41e30b0d5974df3d1b2429fbdf728222a4c

Можно переписать hex в классический вид командой `fromhexaddress` (приглядывайтесь к первой строчке на скрине ниже)

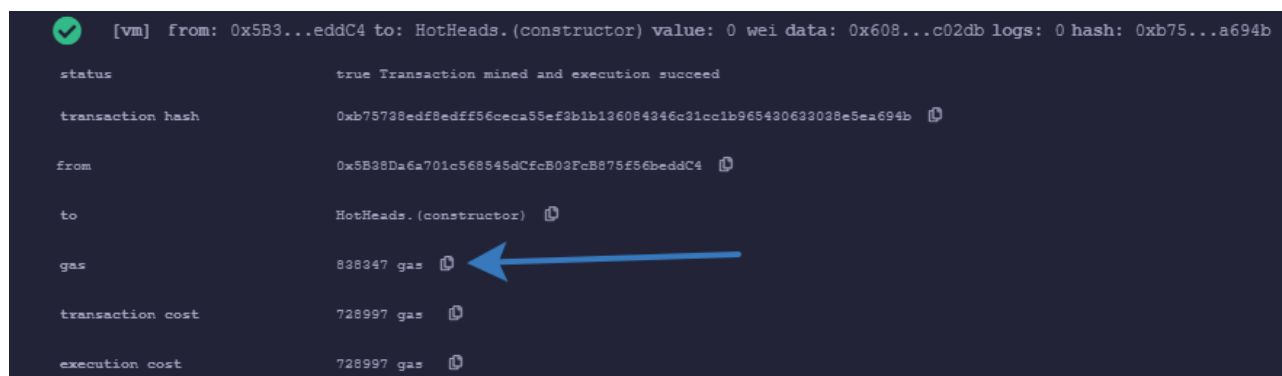
```
21:00:24  ↓  fromhexaddress 184eb41e30b0d5974df3d1b2429fbdf728222a4c
21:00:24  ↑  ZEFnGiHuwDStHnBA3cvAgPPFhhAKKqXQna
```

В веб кошельке, кстати, при отправке какого либо кошелька в смарт в виде данных кроме hex формата надо ещё добавить 0x как в старой доброй классике



При `aidrop` не стоит засовывать в один блок больше 50 транзакций. Иначе тратится вся сумма на балансе. 20 транзакций безопасно.

Если контракт оказывается битым (после его выгрузки в транзакциях нет значка «добыто» и на зероскане в блоке смарт будет отображаться черным), то следует увеличить газ. Рекомендуются ставить на 20% больше, чем показано в ремиксе



Также укажите с какого кошелька будет создан смарт

Опции

Количество	<input type="text"/>	↕	ZHC	▼
Лимит Gas	1000000	↕	Цена Gas	0.00000040
Адрес отправителя	ZEFnGiHuwDStHnBA3cvAgPPFhhAKKqXQna			

ОЧИСТИТЬ ВСЕ ОТПРАВИТЬ КОНТРАКТ

Ниже показаны нормальные ситуации.

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

469e8e0ff1aa688d3de869572cbaca3d2727405e5883877f74cddbfb4453c1	3 confirmations	2022-06-04 22:17:04
ZbFkrDeGDr6EW1UyXBCGV7vSojEgpLF8fm	495.45034960 ZHC	Contract Create
Gas Back	→ eccecb4245cd6ddb4fb1bc4e24a1b8dea1c30e65 ZJWpducUCKwXB6yktUw6FjP615j9oLvQQ	494.00784560 ZHC
Mint Tokens	→ ZbFkrDeGDr6EW1UyXBCGV7vSojEgpLF8fm	0.26905840 ZHC
Mint Tokens	→ ZEFnGiHuwDStnBA3cvAgPPFhhAKKqXQna	999000000 TESTF9
	→ ZTwG2DpQNsRoczDJg4jXcZ43gCwoEXmmsc	1000000 TESTF9
		Fee 1.1734456 ZHC

Узнать необходимый газ можно в ремиксе, развернув данные о транзакции. Но при попытке послать транзакцию, указанную на картинке ниже, с газом 50000 она не прошла, но с газом 100000 прошла. Рекомендуемый газ для любых транзакций в zh 250000, в 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	0xa8f451b9a1b0ba79d452e7eb449f28fda0d6d190667dbf92ab220de9137e
from	0xab8482f64d9c6d1cfcfb49ae677d0311535cb2
to	ZRC20Token.addNewUser(address,uint256) 0xd91450CE3D386E254917e451a84e9943f39138
gas	42005 gas
transaction cost	36526 gas
execution cost	36526 gas
input	0x10f...0000a
decoded input	{ "address": "0xab8109628c481177ec7E3F571ccCaE3A9e22C02db", "uint256": "100" }

Также там можно узнать какая именно команда (код команды) при этом выполняется. Это необходимо для взаимодействия с блокчейном через консоль.

Zh поддерживает последнюю версию солидита (может не работать корректно), но кошелек поддерживает только 0.4.26 версию (т. к. ABI в более поздних версиях другое). В первых контрактах я использовал версию 0.4.18, как в стандартном примере qtum QRC20 Token <https://docs.qtum.site/en/QRC20-Token-Introduce.html>, потому что не изменял стандартное значение газа. А при компиляции на версиях 0.7 даже стандартный пример смарта бьётся если оставить газ 250000.

Для того, чтобы комфортно писать на версии солидита 0.8 необходимо преобразовать abi через питоновский скрипт <https://abi.zhcash.net/>. Тогда можно комфортно запускать все современные проекты солидита на zh и сам код на последних версиях пишется гораздо приятнее.

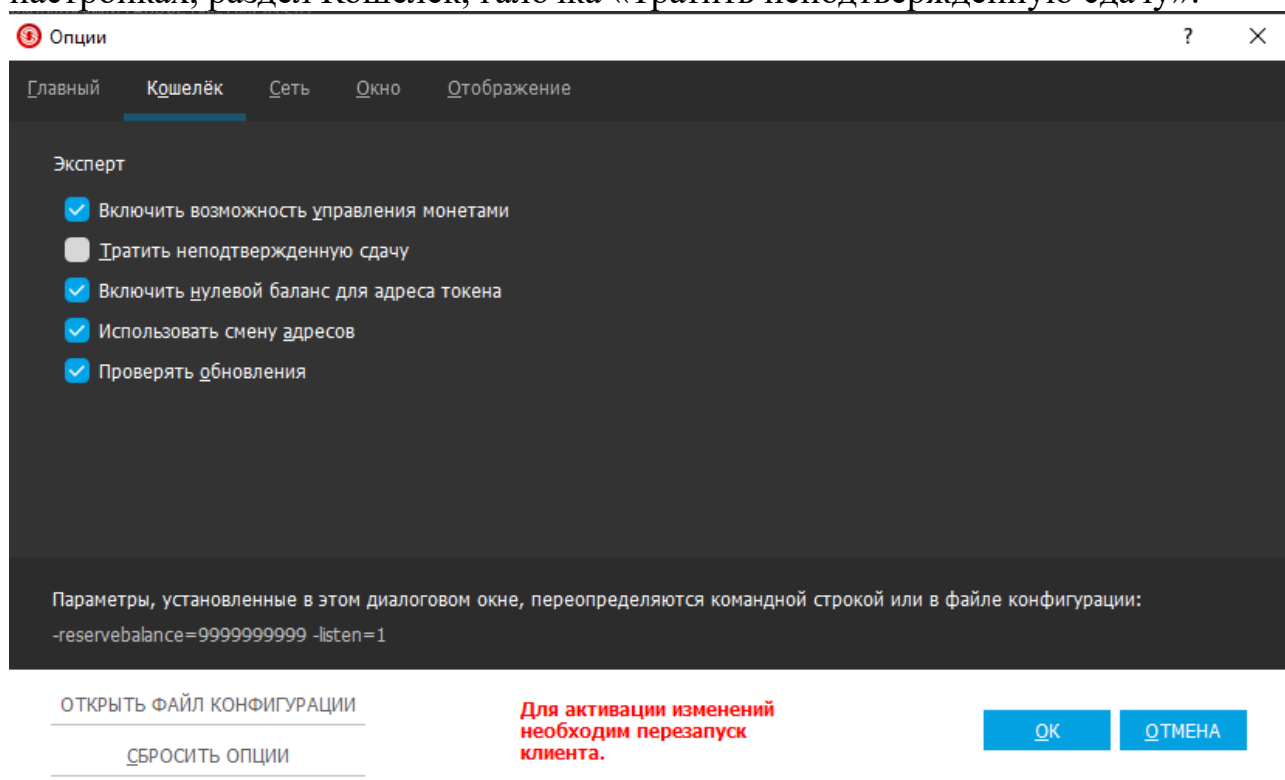
Загружать смарты и отсылать данные (sendtocontract) на смарт рекомендуется с не более 20 отправок за один блок, если у вас установлены значения в настройках кошелька по умолчанию, иначе некоторые транзакции могут

зависнуть, а в них и весь баланс (из за UTXO). Далее будет показано, как в настройках кошелька снять одну галочку и это сразу исправляет данную проблему на 99%. Автор неоднократно с этим сталкивался. Данное правило не касается вызова функций контракта (callcontract) и единичных отправок токенов кому либо. Было замечено, что при попытке впихнуть более 30 транзакций в один блок с одного кошелька начинается сильное списание зх с баланса (от 1 тыс до 400 тыс). Так на дроп можно слить весь баланс ноды (1 млн ), разослав токены 60 пользователям в одном блоке. В итоге автор сделал дроп рассылку 20 пользователям в одном блоке (с задержкой в 10 минут). В час рассылается 200 пользователям, что приемлемо. За 4 часа дроп начилился всем. Слившиеся зх можно вернуть, отменив зависшую транзакцию. Всё таки есть способы обойти это ограничение и начилить токены на тысячи адресов мгновенно (разбив свой кошелек на множество маленьких по одному шекелю, либо используя так называемые «пакетные транзакции» `sendrawtransaction` <https://zh.cash/docs/en/ZHCash-RPC-API/#sendrawtransaction> ), либо написать смарт который будет получать зх и делать отправления по указанным адресам (смарт может делать пакетную транзакцию), но оставим это упражнение читателю.

Связано это с тем, что сдача по алгоритму UTXO должна вернуться в следующем блоке и при этом может израсходоваться весь баланс. Из за UTXO он может перестать отображаться в общем балансе кошелька (транзакция может зависнуть). Так потерялось 400 тыс. шекелей за 5 минут. Можно отменить зависшую транзакцию в любой момент (даже спустя несколько месяцев), тыкнув на «Отказ от транзакции» и после отмены на баланс возвращается потерянная сумма. Эта транзакция отображается под вопросом (вместо галочки слева).

✓	31.07.2022 21:54	Отправка контракта	(ae5fb9d7c1e58d24ebc02ec9272b335f253a45...	-0.11472965
✓	31.07.2022 21:31	Отправка контракта	(ae5fb9d7c1e58d24ebc02ec9272b335f253a45...	-0.19104224
✓	31.07.2022 21:21	Отправка контракта	(ae5fb9d7c1e58d24ebc02ec9272b335f253a45...	-0.16942780
✓	31.07.2022 17:45	Отправка контракта	(ae5fb9d7c1e58d24ebc02ec9272b335f253a45...	-4.28824059
?	31.07.2022 17:37	Отправка контракта	(ae5fb9d7c1e58d24ebc02ec9272b335f253a45...	[-10.08474746]
✓	26.07.2022 16:48	Отправка контракта	(ae5fb9d7c1e58d24ebc02ec9272b335f253a45...	-0.10364800
✓	09.06.2022 12:00	Отправка контракта	(ae5fb9d7c1e58d24ebc02ec9272b335f253a45...	-0.10129600
✓	09.06.2022 11:59	Отправка контракта	(ae5fb9d7c1e58d24ebc02ec9272b335f253a45...	-0.10129600
✓	09.06.2022 11:16	Отправка контракта	(ae5fb9d7c1e58d24ebc02ec9272b335f253a45...	-0.10129600
✓	09.06.2022 10:31	Отправка контракта	(ae5fb9d7c1e58d24ebc02ec9272b335f253a45...	-0.04129600
✓	09.06.2022 10:31	Отправка контракта	(ae5fb9d7c1e58d24ebc02ec9272b335f253a45...	-0.04129600
✓	09.06.2022 10:31	Отправка контракта	(ae5fb9d7c1e58d24ebc02ec9272b335f253a45...	-0.04129600
✓	09.06.2022 10:31	Отправка контракта	(ae5fb9d7c1e58d24ebc02ec9272b335f253a45...	-0.04129600
✓	09.06.2022 10:31	Отправка контракта	(ae5fb9d7c1e58d24ebc02ec9272b335f253a45...	-0.04129600
✓	09.06.2022 10:31	Отправка контракта	(ae5fb9d7c1e58d24ebc02ec9272b335f253a45...	-0.04129600
✓	09.06.2022 10:31	Отправка контракта	(ae5fb9d7c1e58d24ebc02ec9272b335f253a45...	-0.04129600

Для того, чтобы спамить блокчейн оказалось достаточно снять галочку в настройках, раздел Кошелёк, галочка «Тратить неподтвержденную сдачу».

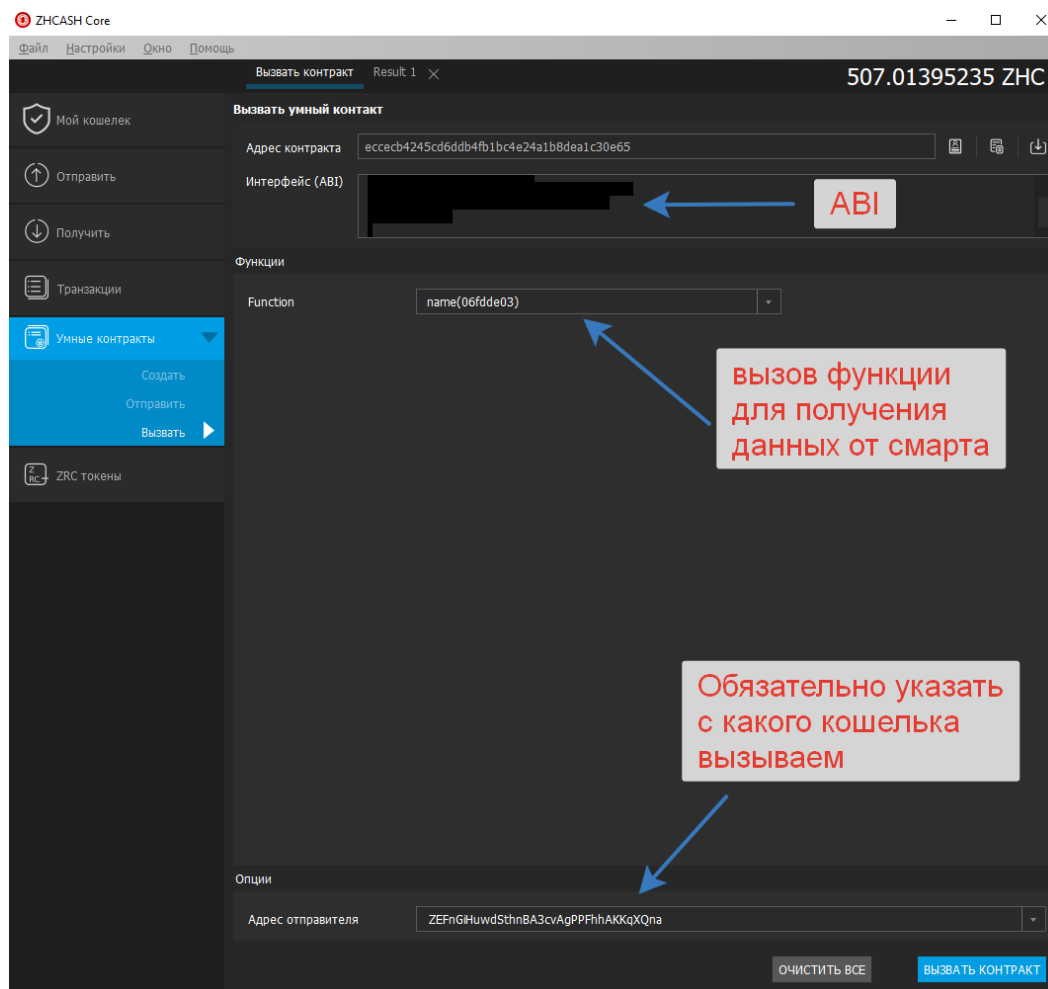


Но все равно при непрерывном спаме может возникнуть ситуация, когда некоторые транзакции зависают со всеми средствами. Пока решение этого вопроса полностью не найдено. Каким то образом надо отменять зависшие транзакции.

Рекорд по спаму засвидетельствован здесь <https://zeroscan.io/block/771654>



Рассмотрим как осуществлять обмен информацией со смартом. Разберем сначала как взаимодействовать со смартом через графический интерфейс кошелька. Затем рассмотрим как это сделать через командную строку.



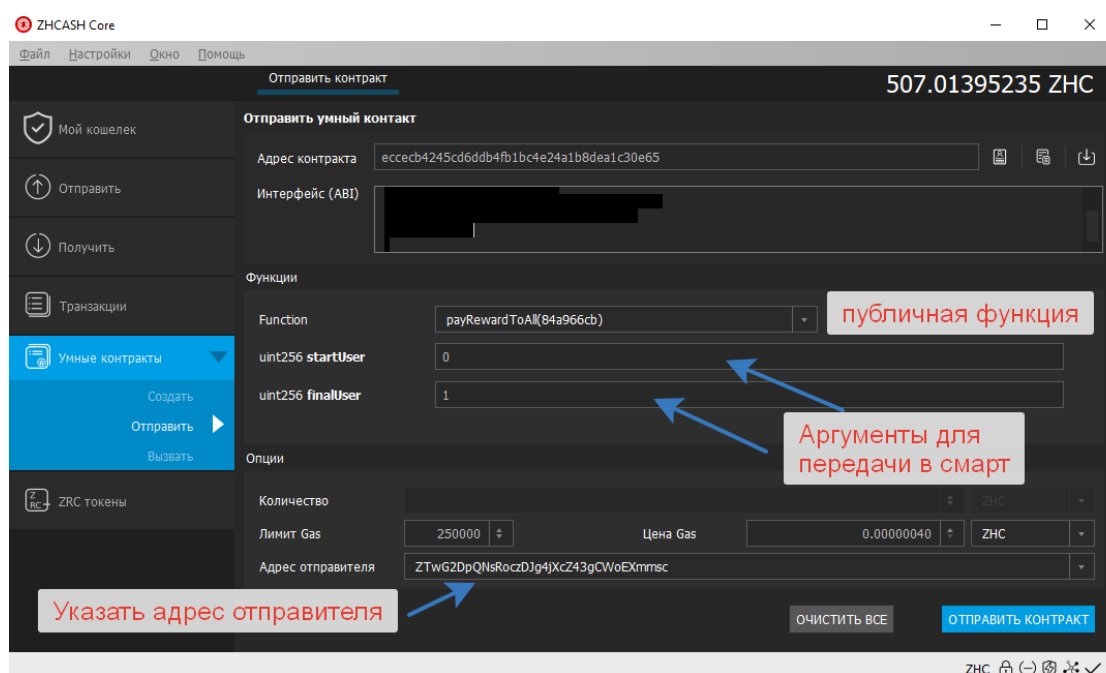
Если мы хотим получить данные, то следует использовать вкладку «Вызвать». Получим следующий результат.

Contract Summary	
ContractAddress	eccecb4245cd6ddb4fb1bc4e24a1b8dea1c30e65
Function	name()
SenderAddress	
Result	
string	TESTF9

Так мы можем получить любое значение публичной переменной или выполнение внешней (external view) функции.



Если мы хотим отправить данные, то следует использовать вкладку «Отправить»



Если мы хотим выплатить вознаграждение всем пользователям, то следует также выплачивать порциями по 20 транзакций в блоке. И поставить большее значение газа. После такого обращения получим следующее.

Contract Summary	
Transaction ID	9972da26f233d7e963aa1ca93eef7bb341c2a89571c2762545d05c92b8f698f8
SenderAddress	ZTwG2DpQNsRoczDJg4jXcZ43gCWoEXmmmc
Hash160	ae5fb9d7c1e58d24ebc02ec9272b335f253a4509

Подождав новый блок и зайдя в «Транзакции» заметим, что появится значок «Добыто»

05.06.2022 16:03	Добыто	(ZL25qnzUA7uz7kQ8YC3vmZAt8jgS19UXwL)	[0.07348320]
05.06.2022 16:02	Отправка контракта	(57884d4449512ede1e192415c8cf4029969fd5f0)	-0.10188400

Это значит, что отправка данных успешна.

Теперь рассмотрим взаимодействие с блокчейном через консоль.

Для этого есть две команды: `callcontract` для получения данных и `sendtocontract` для отправки данных в смарт. Ниже приведены пример использования.

```
callcontract  eccecb4245cd6ddb4fb1bc4e24a1b8dea1c30e65
06fdde03
```

call

[call] from: 0x5B38Da6a701c568545dCfcB03FcB875f6beddC4 to: ZRC20Token.name() data: 0x06f...dde03

from

0x5B38Da6a701c568545dCfcB03FcB875f6beddC4

to

ZRC20Token.name() 0xd9145CCE22D366254917e401e84e9942F39138

execution cost

21683 gas (Cost only applies when called by a contract)

input

0x06f...dde03

decoded input

{

decoded output

{ "0": "asking: TESTFLO" }

logs

{ }

команда

```
callcontract ecceb4245cd6ddb4fblbc4e24alb8dealc30e65 0x06fddde03
{
  "address": "ecceb4245cd6ddb4fblbc4e24alb8dealc30e65",
  "executionResult": {
    "gasUsed": 21046,
    "excepted": "Revert",
    "newAddress": "ecceb4245cd6ddb4fblbc4e24alb8dealc30e65",
    "output": "",
```

[illegible]

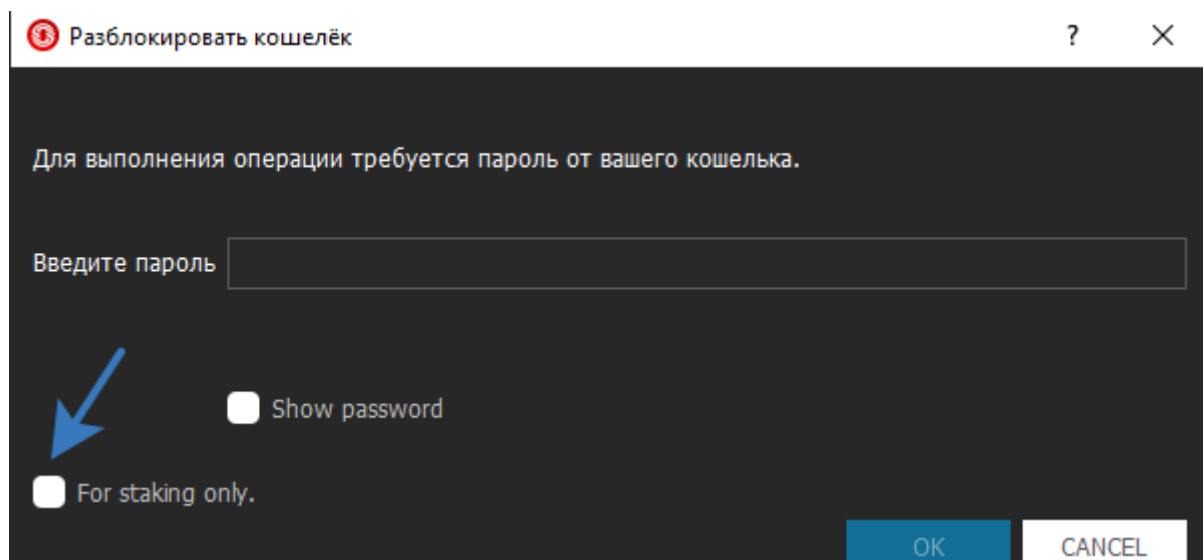
Мы получим какие то данные, которые затем можно перегнать в строку. Ниже показан скриншот для `sendtocontract`

[illegible]

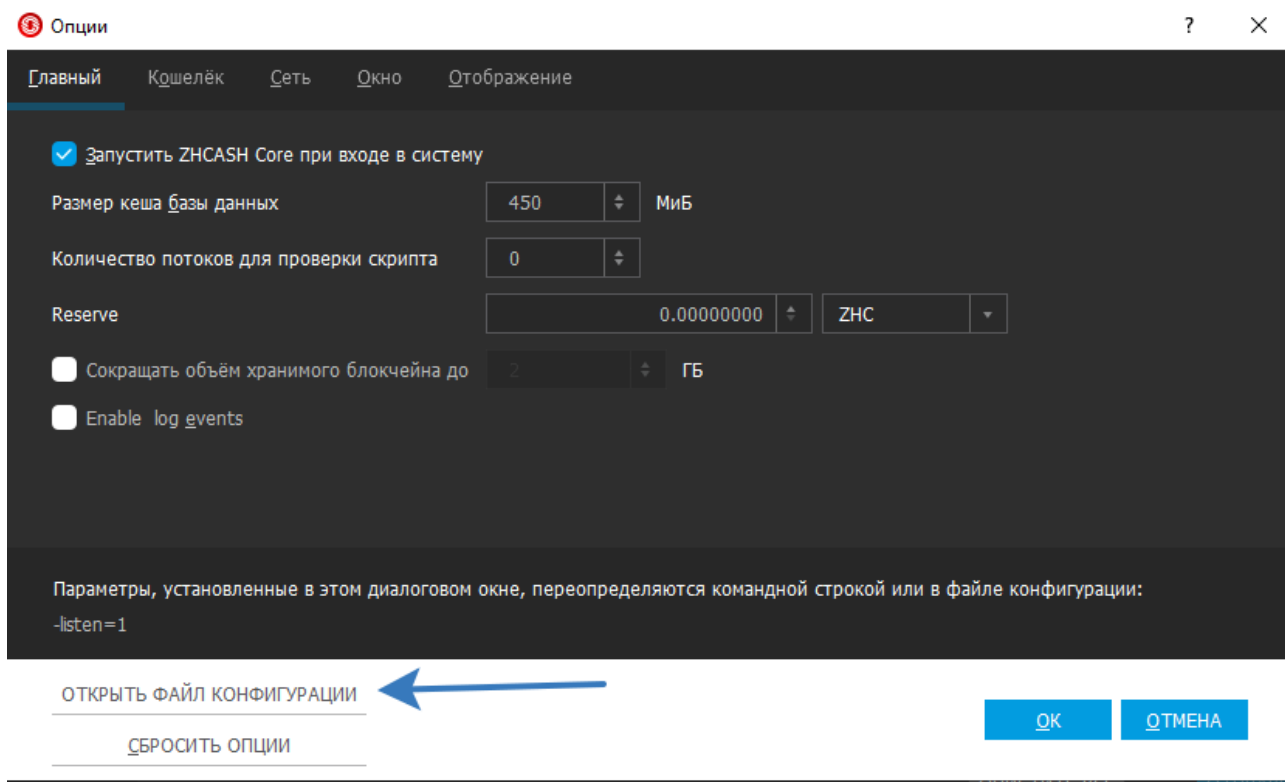
Для того, чтобы взаимодействовать через командную строку `cmd` нужно скачать консольную (серверную) версию кошелька



Затем разблокировать кошелек и снять галочку “For staking only”




Добавить файл конфигурации в параметрах



И сохранить следующий текст, где в последнем аргументе rpcpassword установить свой пароль от кошелька

```
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=[REDACTED]
```



```
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
```

```
#rpcallowip=17.2.7.11
rpcallowip=127.0.0.1
rpcuser=zerohour-rpcuser
rpcpassword=1123581321
```

Для отключения стейкинга недостаточно установить `staking` в `0`,  
нужно также установить `reservebalance=999999999`.

После этих манипуляций можно взаимодействовать с блокчейном через командную строку, что позволяет писать скрипты на питоне для автоматизации каких то действий с блокчейном. Например, для организации дропа токенов, отправки транзакций, создание миллиона тестовых токенов, спама, хакинга и тому подобного. Ниже показано как можно вызвать функцию `callcontract` через `cmd`

[illegible]

## Тестирование смарта

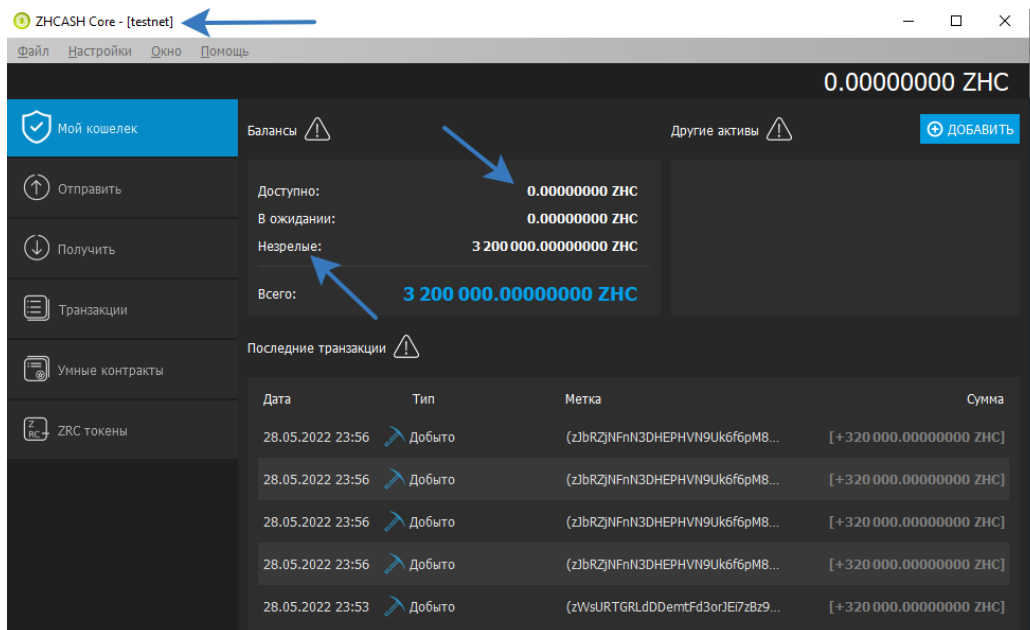
Для тестирования смарта можно использовать три подхода.

1) Запустить свой кошелек zh в режиме тестнет (с ключом -testnet через командную строку).

 Windows PowerShell

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

В данном методе у вас организуется локальный блокчейн с нулевого блока, где вам ещё надо нагенерить 500 блоков для того, чтобы получить тестировочные шекели. Пока вы это не сделаете все шекели будут незрелыми и оплатить создание смарта не получится(



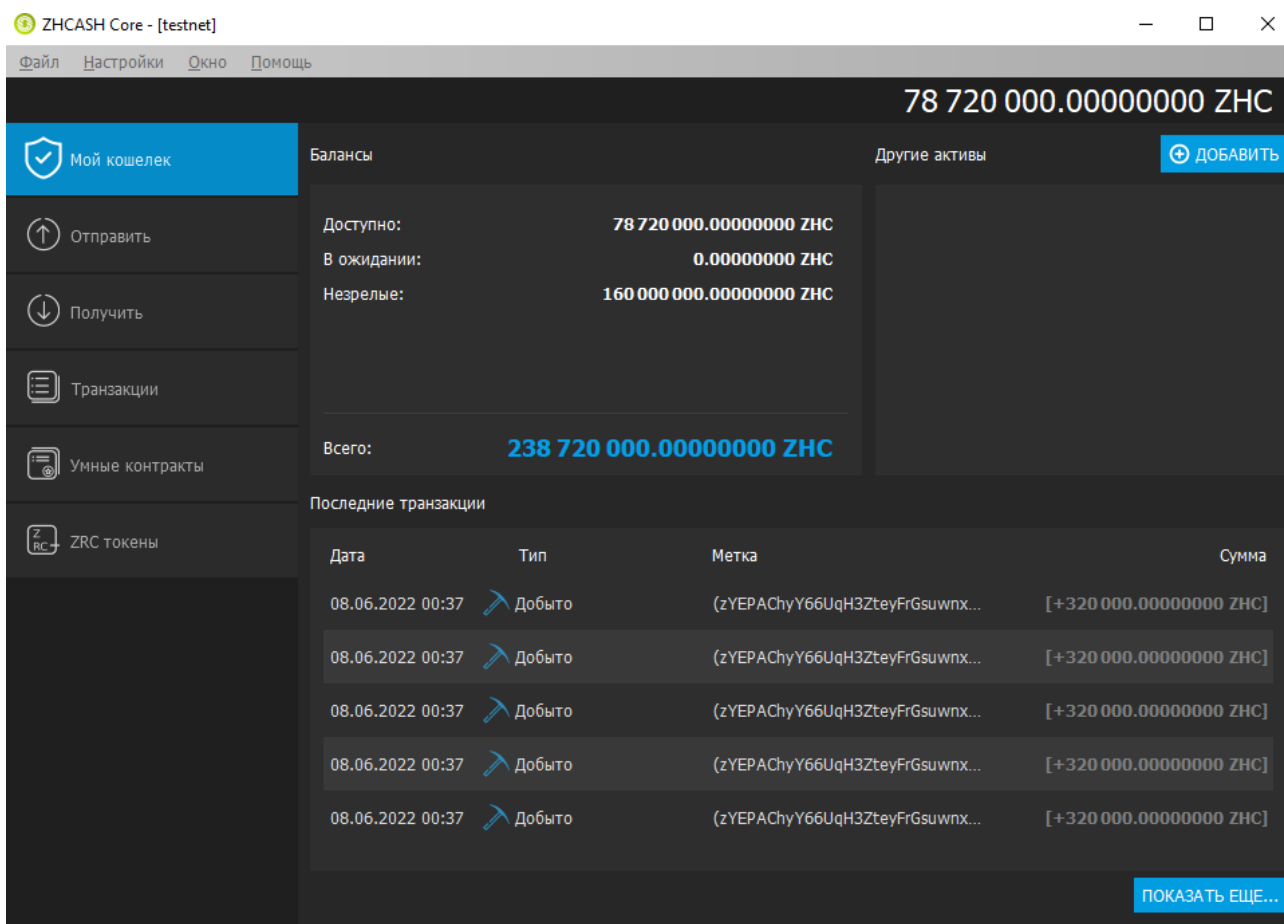
Заходим в консоль, пишем generate 10 и получаем ошибку. Для того, чтобы нагенерить блоки надо запускать с ключами -testnet -deprecatedrpc=generate. Повторяем ещё раз.

```
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> █
```

Тыкаем в консоле эту команду много-много раз, пока что-то не произойдет с балансом в разделе «Доступно». От незрелых шекелей у блокчейна несварение.

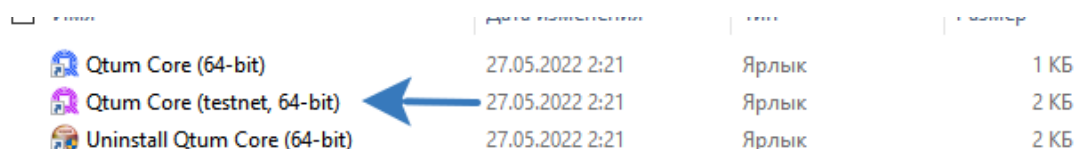


После многократного изнасилования клавиши вверх (для повторения прошлой команды) и enter блокчейн раздупляется и готов к работе



Можно грузить смарты, все проверять, потом генерить блок и все тестить.

2) Использование тестовой сети QTUM. У них есть кран для тестовой сети, где можно указать свой адрес и вам начислят  $50 \pm 20$  тестировочных монет. При скачивании кошелька сразу доступен отдельный кошелек testnet.



На синхронизацию с тестировочным блокчейном уходит порядка 2 часа. Гайд по тестнету qtum <https://docs.qtum.site/en/Testnet-User-Guide.html>

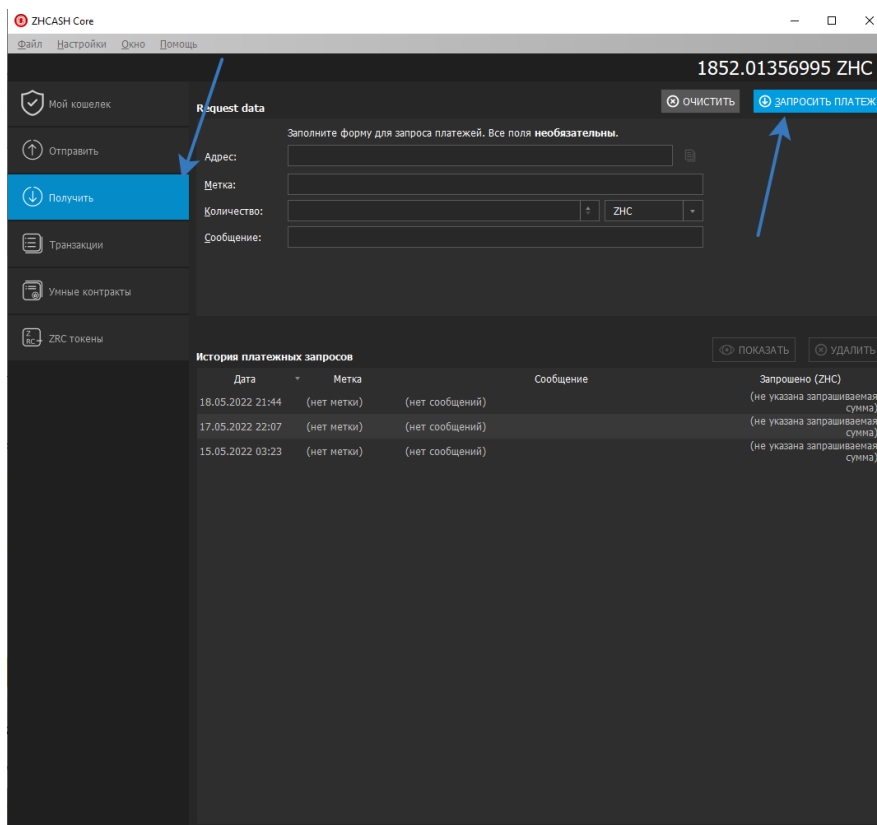


3) Создание десятков тестовых смартов в основной сети (как сделал по началу автор. Поэтому он решил написать гайд), но это осуждается. Ниже приведен результат третьего подхода в тестировании.

☒ 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)   ☐ QRC TEST6 (QRC6)   ☐ ZRC TEST10 (ZRC10)   ☐ QRC TEST6 (QRC6)   ☐ ZRC TEST 2 (ZRC2)   ☐ TEST FINAL (TESTF)   ☐ TEST FINAL 4 (TESTF4)  
☐ 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)   ☐ 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)

## Получение адреса кошелька

Для получения своего номера кошелька надо зайти во вкладку «Получить» и нажать кнопку «ЗАПРОСИТЬ ПЛАТЕЖ»





## PAYMENT INFORMATION

URI: <zerohour:ZRhrbbXyi7DeZkXh63yKZR2waCUYMTvk2w>

Адрес: ZRhrbbXyi7DeZkXh63yKZR2waCUYMTvk2w

КОПИРОВАТЬ URI

СОХРАНИТЬ ИЗОБРАЖЕНИЕ...

КОПИРОВАТЬ АДРЕС

CLOSE

Для того, чтобы снова скопировать номер кошелька следует выделить строку в истории платёжных запросов и скопировать появившийся Адрес

ZHCASH Core

1852.01356995 ZHC

Мой кошелек

Отправить

Получить

Транзакции

Умные контракты

ZRC токены

Request data

Заполните форму для запроса платежей. Все поля **необязательны**.

Адрес: ZRhrbbXyi7DeZkXh63yKZR2waCUYMTvk2w

Метка:

Количество: 0.00000000 ZHC

Сообщение:

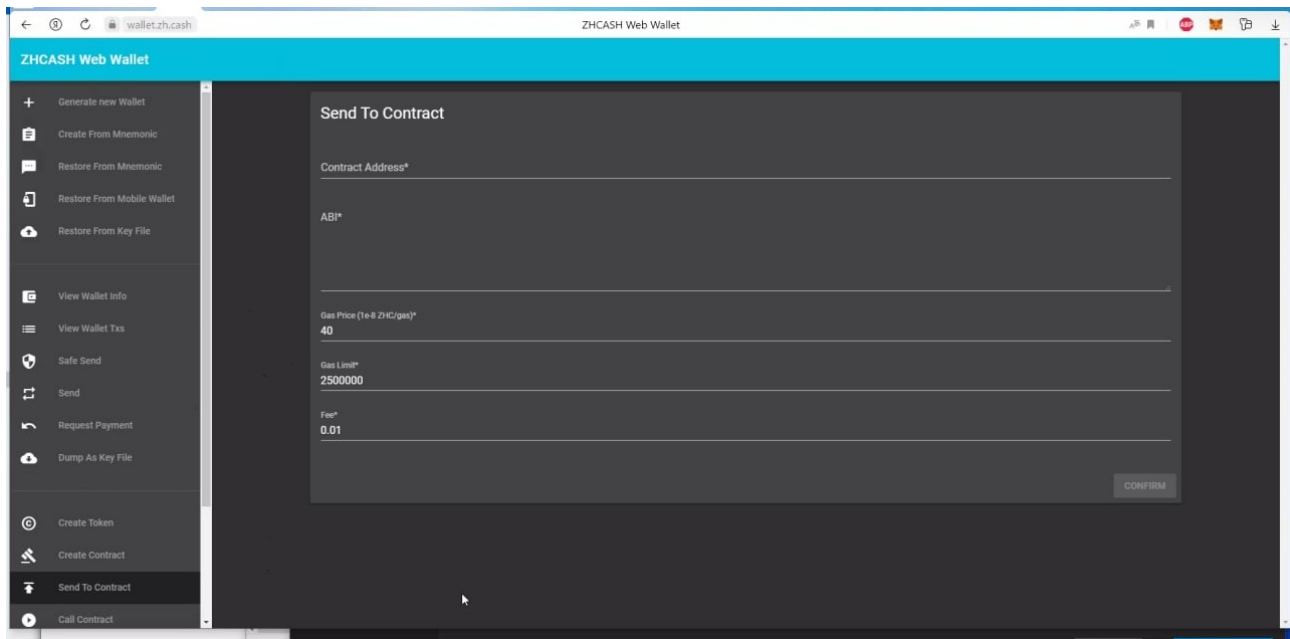
ОЧИСТИТЬ ЗАПРОСИТЬ ПЛАТЕЖ

История платёжных запросов

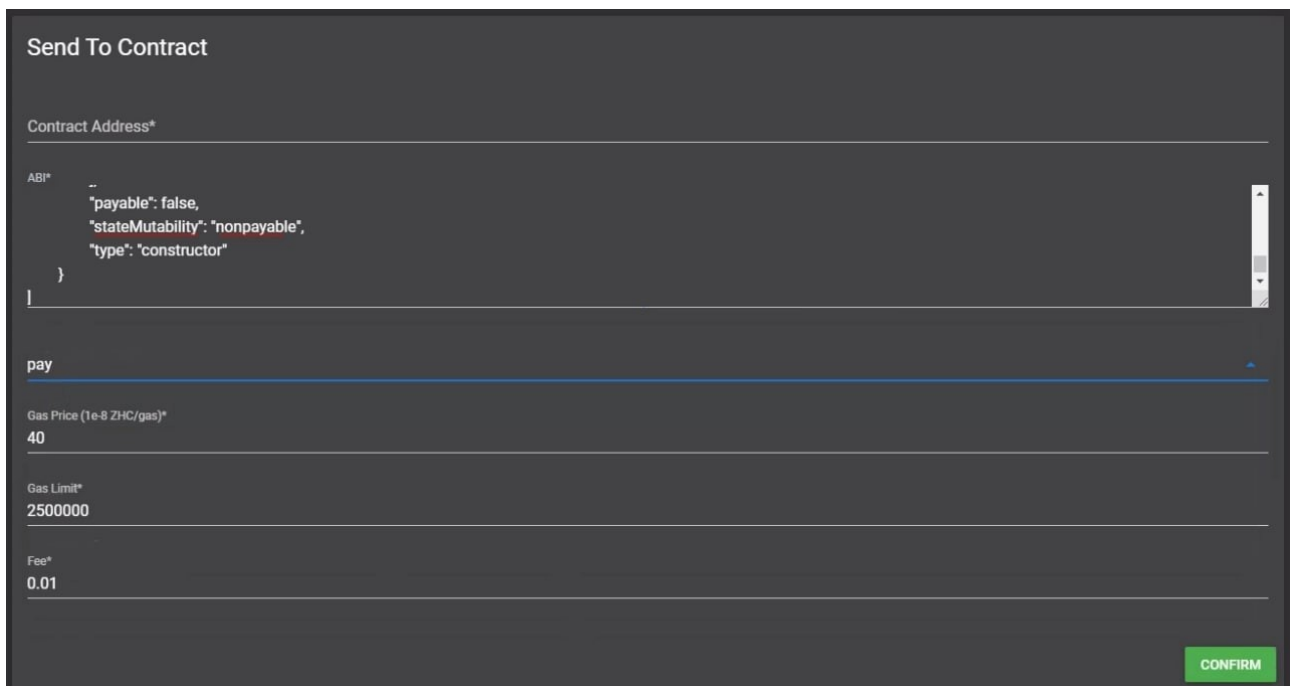
ПОКАЗАТЬ УДАЛИТЬ

Дата	Метка	Сообщение	Запрошено (ZHC)
26.07.2022 16:35	(нет метки)	(нет сообщений)	(не указана запрашиваемая сумма)
18.05.2022 21:44	(нет метки)	(нет сообщений)	(не указана запрашиваемая сумма)
17.05.2022 22:17	(нет метки)	(нет сообщений)	(не указана запрашиваемая сумма)
15.05.2022 03:23	(нет метки)	(нет сообщений)	(не указана запрашиваемая сумма)

# ВЕБ КОШЕЛЕК



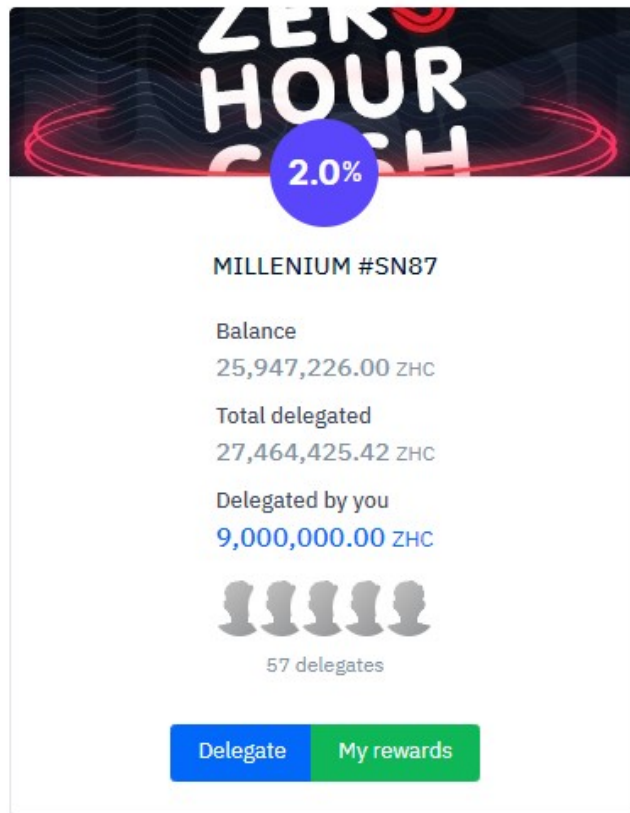
На веб версии нельзя отправить зх вместе с командой sendtocontract



И вообще нельзя отправить зх на смартконтракт. Пока такая функция не добавлена (на момент 21.09.2022).

В остальном веб-кошелек имеет тот же функционал, что и нода.

Например, на 87 ноду **MILLENIUM** <https://zhcash.org/invests?mode=light&page=7>



## Ошибки RPC:

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

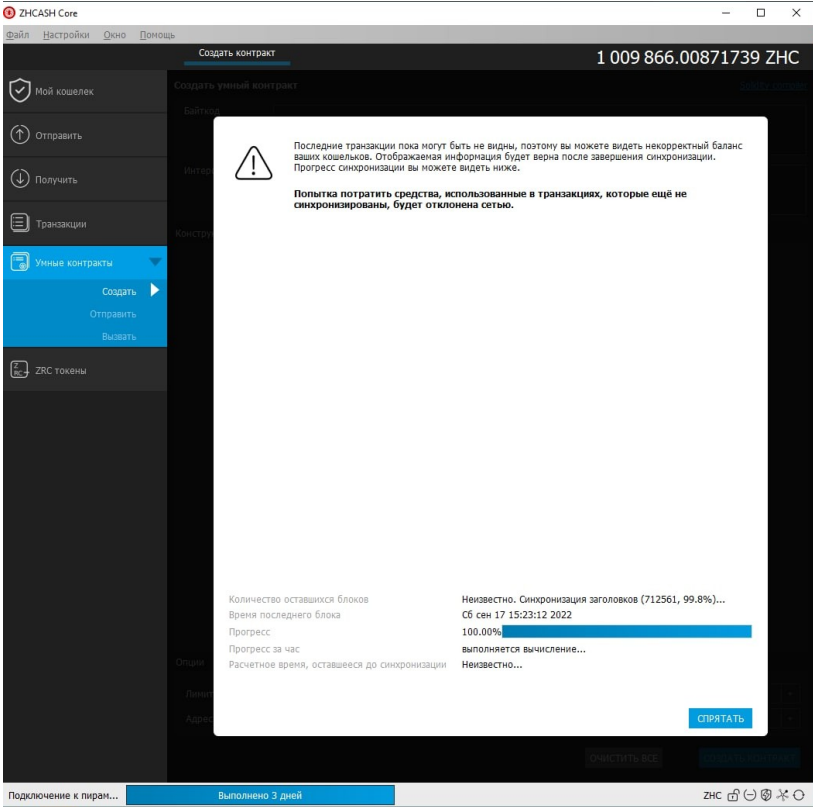
Вы указали в команде не свой номер кошелька

[illegible]

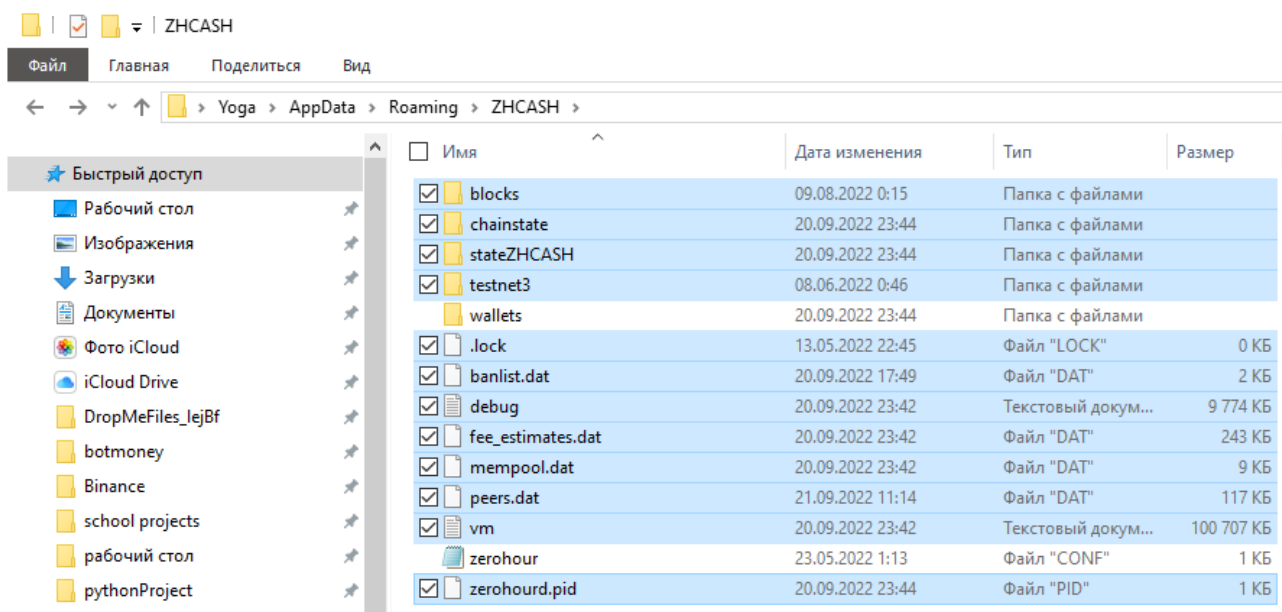
Вот так будет правильно:

[illegible]

Если в какой то момент на жестком диске закончится свободное место, то блокчейн может перестать обновляться



Для начала новой синхронизации надо удалить все файлы из папки ZHCASH (та же папка, где лежит файл конфигурации), кроме папки wallets и файла конфигурации zerohour



Для этого надо закрыть qt кошель и запустить уже после удаления.

## RAW TRANZACTIIONS (сырые транзакции)

В зх используются биткоин raw транзакции [https://bits.media/raw\\_transactions/](https://bits.media/raw_transactions/)  
 Для создания сырой транзакции можно воспользоваться этим гайдом <https://syncagio.medium.com/programmatically-creating-and-signing-a-raw-bitcoin-transaction-using-python-on-a-mac-dffbb7283616>  
 и методами RPC зх <https://zh.cash/docs/en/ZHCash-RPC-API/#rawtransactions>  
 также можно воспользоваться наработками адаптированного питон скрипта

#<https://syncagio.medium.com/programmatically-creating-and-signing-a-raw-bitcoin-transaction-using-python-on-a-mac-dffbb7283616>

```
import subprocess
import json
import pprint
#ZXmoyeS2oYXe1R8ygSJd3yRQRU9o37DDzq
cli = 'zerohour-cli.exe'
r = subprocess.check_output([cli, 'listunspent'])
list = json.loads(r)
i = 0
pprint.pprint(list)
for o in list:
    print(i, o[u'amount'], o[u'address'])
    i = i + 1
id = input("select number of transaction output to spend: ")
t = list[int(id)][u'txid']
v = list[int(id)][u'vout']
a = list[int(id)][u'amount']
receive = input("input receiving address: ")
amount = input("input amount to spend in BTC: ")
fee = float(input("input fee (0.00010000 = about $5): "))
#changeaddress = subprocess.check_output([cli, "getnewaddress"])
#changeaddress = changeaddress[0:len(changeaddress) - 1]
changeaddress = "ZDM1mHyiysN1bmaAvHLZcGuwPf8gxVoF2"
changeamount = float(a) - float(amount) - fee
print("=====transaction details:")
print("transaction is " + t + ", vout is " + str(v))
```

```

print("incoming amount is " + str(a))
print("receive address is " + str(receive))
#print("change address is " + str(changeaddress))
print("=====amounts details:")
print("{:.8f}".format(float(a)) + " incoming amount = ")
print("{:.8f}".format(float(amount)) + " outgoing amount")
print("{:.8f}".format(float(changeamount)) + " change amount")
print("{:.8f}".format(float(fee)) + " fee")
c1 = "[{"txid\":\"" + str(t) + "\",\"vout\":\"" + str(v) + "\"}] "
c2 = "[{\"" + str(receive) + "\":\"" + str(amount) + "\"} \", {\"" + str(changeaddress) + "\":\"" + str(changeamount) + "\"}]"
#passphrase = input("enter your wallet passphrase: ")
print(c1)
print(type(c2))
try:
    tx = subprocess.check_output([cli, "createrawtransaction", c1, c2])
except subprocess.CalledProcessError as e:
    print("createrawtransaction error")
print("tx:",tx)
tx = tx[0:len(tx) - 2]
print("tx:",tx)

'''
try:
    pp = subprocess.check_output([cli, "walletpassphrase", passphrase, "100"])
except subprocess.CalledProcessError as e:
    print("walletpassphrase error")
'''

#dumpprivkey ZDM1mHyiysN1bmaAvHLZcGuwPf8gkxVoF2

privkey = "[\"L1tNd4DgsrHm6wDBuAvu3WszY78QXqzNf3zKSZPzQLz1ioWtKpZG\"]"

try:
    tx = subprocess.check_output([cli, "signrawtransactionwithkey", tx, privkey])
except subprocess.CalledProcessError as e:
    print("signrawtransactionwithkey error")
'''
try:
    tx = subprocess.check_output([cli, "signrawtransactionwithwallet", tx])
except subprocess.CalledProcessError as e:
    print("signrawtransactionwithwallet error")
'''

stx = json.loads(tx)
print("=====signed transaction:\n")
print(stx[u'hex'])

```

## API zeroscan

У зероскана есть апи, по которому можно вытащить любую инфу из блокчейна. Например, все транзакции по какому-нибудь кошельку

<https://ws.zerescan.io/address/ZEFnGiHuwSthnBA3cvAgPPFhhAKKqXQna/basic-txs>

Описание апи аналогично кутумовскому

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



## Установка ноды ZHCASH на серверную версию Ubuntu с версии 18.04 и выше.

Рекомендую запускать обновление блокчейна ноды в сессии tmux  
<https://losst.pro/shpargalka-po-tmux>

Для подключения по ssh рекомендую MobaXterm

0. Заходим под суперюзером  
`sudo su`

Устанавливаем tmux  
`apt install tmux`

<https://losst.ru/shpargalka-po-tmux>

в root создаем папку ZHCASH, где все будет крутиться  
заходим в эту папку  
`cd /root/ZHCASH/`

1. Скачиваем архив с бинарниками и распаковываем его:

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

`chmod -R 777 ./`

Ставим архиватор unzip (по умолчанию есть в убунте), если не разархивировалось в папку ZHCash-Console-Linux: `apt install unzip`

После разархивирования появится папка info-read-me в который лежит одна из версий текущего гайда.

/root/ZHCASH/ZHCash-Console-Linux/info-read-me/ ▾	
▼ Name	Size (KB)
..	
Chinese-information-from-the...	1
Deutsch-information-from-the...	1
English-information-from-the-...	1
PDF Guide_ZHCASH_eng.pdf	185
Hindi-information-from-the-de...	1
Korean-information-from-the-...	1
Russian-информация-от-раз...	1
Spanish-information-from-the...	1
Turkish-information-from-the-...	1
PDF Гайд_ZHCASH_rus.pdf	591

2. Даем права на исполняемые файлы для пользователя:

```
chmod u+x zerohour && chmod u+x zerohour-cli
```

3. Создаем папку данных кошелька и в ней файл конфиг где прописываем параметры работы демона в фоновом режиме и включение режима staking в кошельке:

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

Запустится текстовый редактор nano, пропишем там 2 параметра:

```
daemon=1
```

```
staking=1
```

после этого сохранить файл конфигурации клавишами CTRL+O, выйти из редактора nano CTRL+X

4. Создаем сессию в tmux под именем zh

```
tmux new-session -s zh
```

5. Запускаем в этой сессии tmux ноду:

```
./zerohourd
```

Появится надпись, что ZHCASH стартовал. Оставляем сервер на пару часиков. Далее кошелек начнет загружать блоки. Посмотреть сколько уже скачано можно подключившись к серверу по новой (не в той сессии tmux, где происходит обновление ноды):

```
./zerohour-cli getblockchaininfo | grep blocks
```

Когда когда количество блоков сравняется с последним блоком в эксплорере zhcash, значит кошелек успешно синхронизировался и готов к стейкингу.

Для отключения стейкинга (актуально для бирж и если вам необходимо иметь все зх на балансе) недостаточно установить `staking = 0`, нужно также установить `reservebalance=9999999999` в конфиге `zerohour.txt`

## Остановить ноду на сервере

```
./zerohour-cli stop
```

## Для запуска надо запустить

```
./zerohourd
```

## Лайфхаки для Питона

Установить пип на убунте

```
sudo apt -y install python3-pip
```

## Отключить флask

```
from sys import platform
```

```
try:
```

```
    if platform == "linux" or platform == "linux2":
```

```
        list_files = subprocess.run(["lsof", "-i:5000"], stdout=subprocess.PIPE, text=True)
```

```
        list1 = list_files.stdout.split()
```

```
        pid_index = list1.index('python3') + 1
```

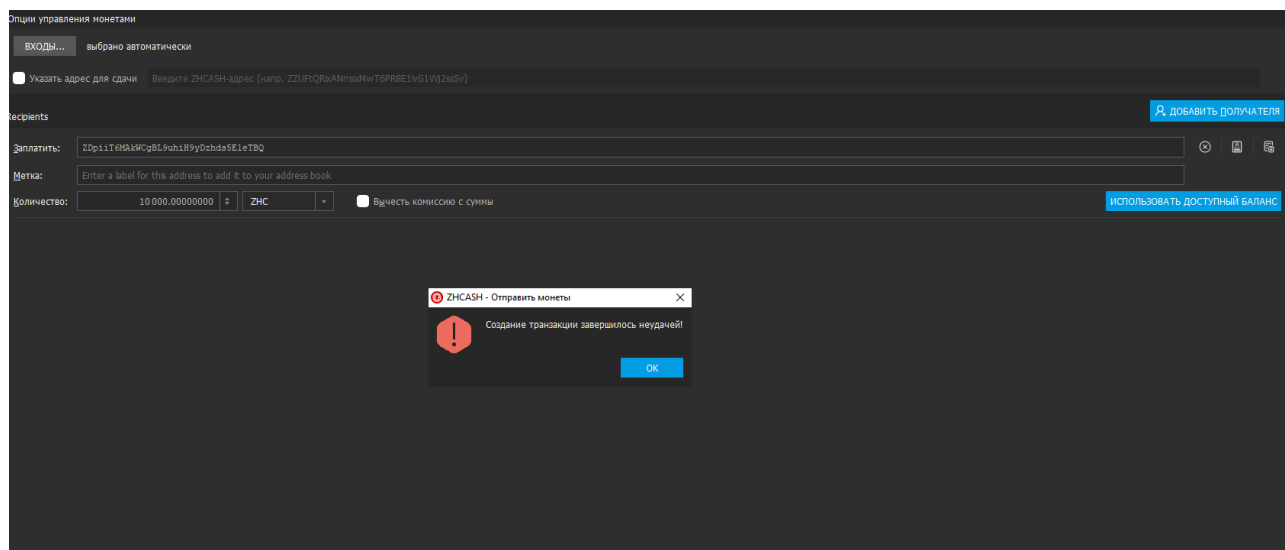
```
        pid = str(list1[pid_index])
```

```
        list_files2 = subprocess.run(["kill", "-9", pid], stdout=subprocess.PIPE, text=True)
```

```
except Exception as e:
```

```
    print(e)
```

**Если создать слишком много входов, то можно получить такую ошибку**



Для обхода этой ошибки — выберите один из входов (слева наверху)

## Заключение

Автором был написан смарт для токена LIFT. Там же находится последняя актуальная версия гайда.

<https://github.com/dimaystinov/Token-LIFT-ZHCASH>

Автор выражает благодарность инициаторам создания токена LIFT

[https://t.me/lift\\_club](https://t.me/lift_club) с адресом

f180d0a911d09853685764a9ad6d366398c50656

Николаю, Арджуну и Денису.

Главному инженеру блокчейна зх Роману Гравитону, программисту Alex, разработчику Mike Gurov за ответы на тупые вопросы, которые легли в основу данного гайда.

@QtumLeandro (Из чата <https://t.me/qtumofficial> ) за ответ, что отправлять данные в смарт надо всё-таки командой sendtocontract.

Раулю @kt2090 за гайд по установке ноды на сервере по ssh

Николаю Ерёменко за помощь в сопровождении написания смартов, программ и данного гайда.

Витале за проект Donation system

Проекту видеосервиса Connect <http://qtconnect.io>

Сообществу Zero Gravity Foundation

Каждый проект способствовал улучшению данного гайда

Донаты принимаются в зх на кошель:

ZEFnGiHuwDStHnBA3cvAgPPFhhAKKqXQna