

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 смарт на месяц раньше

Автор

Определения:

- 1) zh – блокчейн 5го поколения ZHCASH. Результат скрещивания (любви) биткойна и эфира, прошедший генные модификации <https://zh.cash/> . Произносится как «зх». Является противоположностью «хз».
- 2) Шекель — основная монета ZHC блокчейна ZHCASH.
- 3) Закинь шекелей — перечисли на мой кошелек ZHC
- 4) Закинь лифта – перечисли на мой кошель токенов 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/>
- 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 смарт 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 смарт.

10) Ремикс – эфировская среда для разработки смарт <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/>

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

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

Система 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 это прописывать адрес следует в формате hex

Создать умный контракт

Solidity compiler

Байткод

```

ffffffff1673ffffffffffffffffffffffffffffffff168152002001908152002001000020019055308173
ffffffff168373ffffffffffffffffffffffffffffffff167fddf252ad1be2c89b69c2b068fc378daa952ba7f163c4a11628f55a4df523b3
ef836040518082815260200191505060405180910390a3505050565b600080828402905060008414806123da57508284
828115156123d757fe5b04145b15156123e257fe5b80915050929150505600a165627a7a72305820efd2921a4b9f831da2
873cddd3aa62b76c86c684bb0d0732da31fc4b5292c030029

```

Интерфейс (ABI)

Конструктор

address_firstUser

ae5fb9d7c1e58d24ebc02ec9272b335f253a4509

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

```
15:23:20 184eb41e30b0d5974df3d1b2429fbbf728222a4c
15:23:20 gethexaddress ZEFnGiHuwdSthnBA3cvAgPPFhhAKKqXQna
```

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

Contract Summary	
ContractAddress	0a2b496fa9ea27e3d2950e9d1b5056bb751ae3eb
Function	getMyAddress()
SenderAddress	
Result	
address	184eb41e30b0d5974df3d1b2429fbd728222a4c

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

```
21:00:24  ⬇  fromhexaddress 184eb41e30b0d5974df3d1b2429fbdf728222a4c
21:00:24  ⬆  ZEFnGiHuwDStHnBA3cvAgPPFhhAKKgXQna
```

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. Ниже показаны нормальные ситуации.

04.06.2022 22:16	Добыто	(ZbFkrDeGDr6EW1UyXBCGV7vSojEgplLF8fm)	[0.26905840]
04.06.2022 22:13	Отправка контракта	(fea82863035a4a5edf5fe8434a39b17e5ab22a05)	-1.44250400

469e8e0ff1aa6888d3de869572cbaca3d2727405e5883877f74cddbfb4453c1		3 confirmations	2022-06-04 22:17:04
ZbFkrDeGDr6EW1UyXBCGV7vSojEgplLF8fm	495.45034960 ZHC	→ eccecb4245cd6ddb4fb1bc4e24a1b8dea1c30e65 ZJWpducUCKwnXB6yktUw6FjP615j9oLvQQ	Contract Create 494.00784560 ZHC
Gas Back		→ ZbFkrDeGDr6EW1UyXBCGV7vSojEgplLF8fm	0.26905840 ZHC
Mint Tokens		→ ZEFnGiHwudStbnBA3cvAgPPFhhAKKqXQna	999000000 TESTF9
Mint Tokens		→ 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	0xa5f451b8a1b8bca796d52eefeb849f28f6ad66d5190667dbf92ab220de917e	
from	0xab8443f64d9c641c0f2b9493a677d0315835cb2	
to	ZRC20Token.addNewUser(address,uint256) 0xd9145CCE52086E254917e481a84e9942f9138	
gas	42005 gas	← минимально необходимый газ
transaction cost	30326 gas	
execution cost	24526 gas	
input	0x10f...0000a	← команда для командной строки
decoded input	{ "address_uint": "0xab8209928c481177e0728f971c0e2839e22C02db", "uint256_value": "10" }	

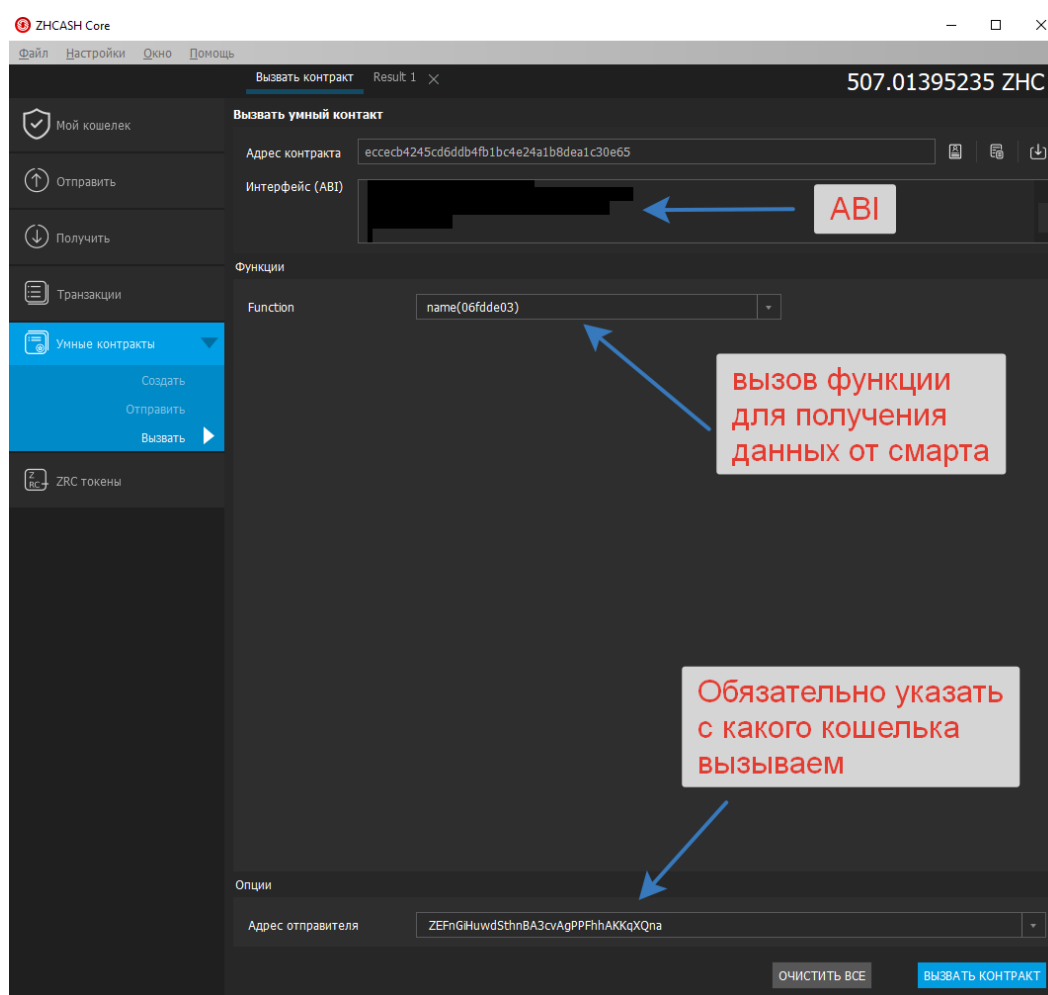
Также там можно узнать какая именно команда (код команды) при этом выполняется. Это необходимо для взаимодействия с блокчейном через консоль.

Zh поддерживает последнюю версию солидیتی, но рекомендуется использовать 0.7 версии. 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 смарт бьётся если оставить газ 250000.

Загружать смарты и отсылать данные (`sendtocontract`) на смарт рекомендуется с кошелька, на котором баланс не больше 100 шекелей, т.к. велика вероятность проёба всего баланса. 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 шекелей с баланса (от 1 тыс до 400 тыс). Так на airdrop можно слить весь баланс ноды (1 млн и больше влѣгкую), разослав токены 60 пользователям в одном блоке. В итоге автор сделал airdrop рассылку 20 пользователям в одном блоке (с задержкой в 10 минут). В час рассылается 200 пользователям, что приемлемо. За 4 часа airdrop начилился всем. Всѣ таки есть способы обойти это ограничение и начилить токены на тысячи адресов мгновенно, но оставим это упражнение читателю.

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.

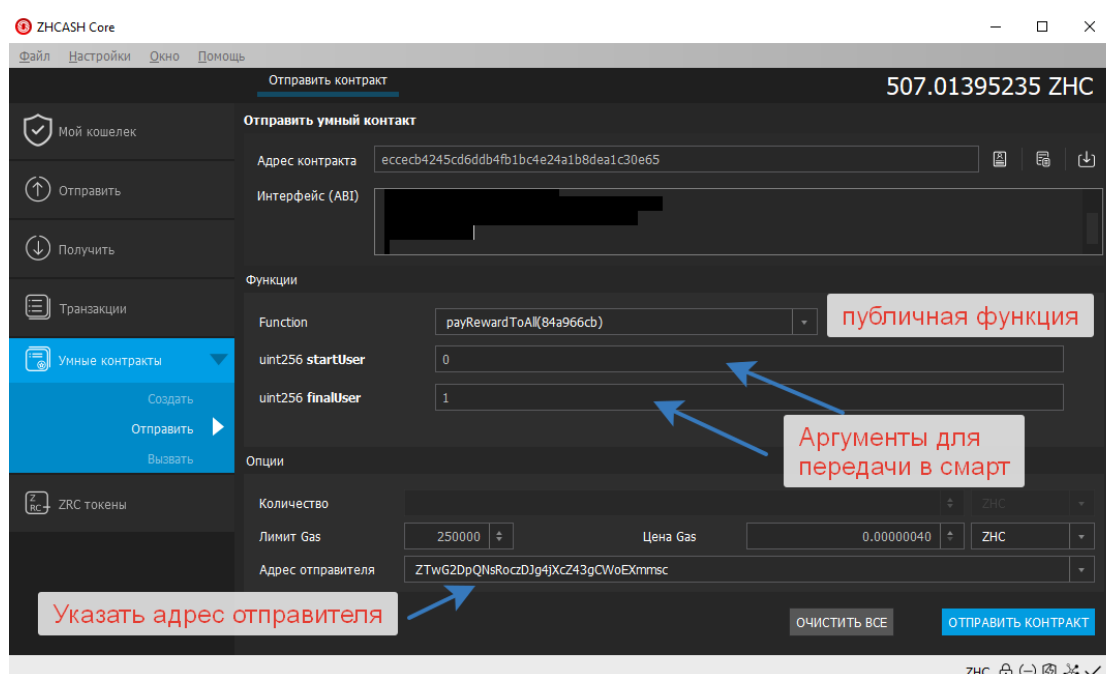


Если мы хотим получить данные, то следует использовать вкладку «Вызвать». Получим следующий результат.

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

Это значит, что отправка данных успешна.

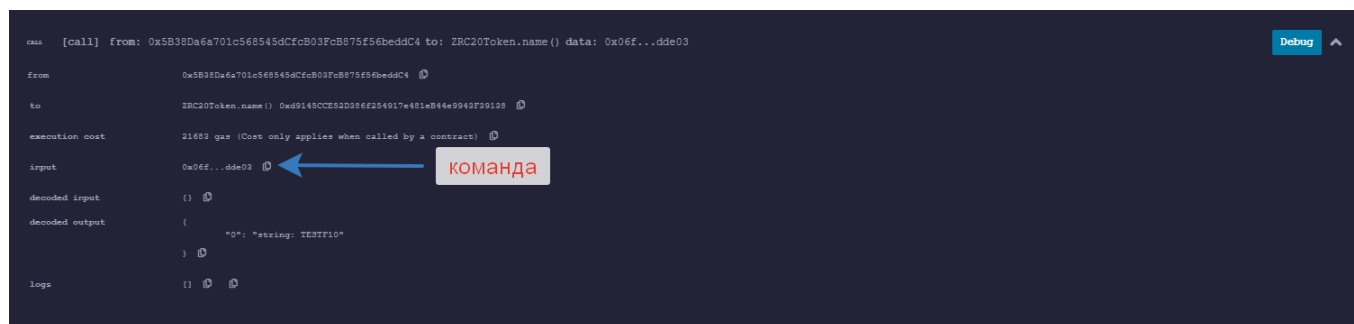
Теперь рассмотрим взаимодействие с блокчейном через консоль.

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

[illegible]

```
callcontract  eccecb4245cd6ddb4fb1bc4e24a1b8dea1c30e65
06fdde03
```

Команду для транзакции через консоль можно получить из ремикса



В названии команды не следует писать 0x. В zh это неправильно. Так вы ничего не получите. Следует писать то, что идет после 0x.

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

Вот так будет правильно:

[illegible]

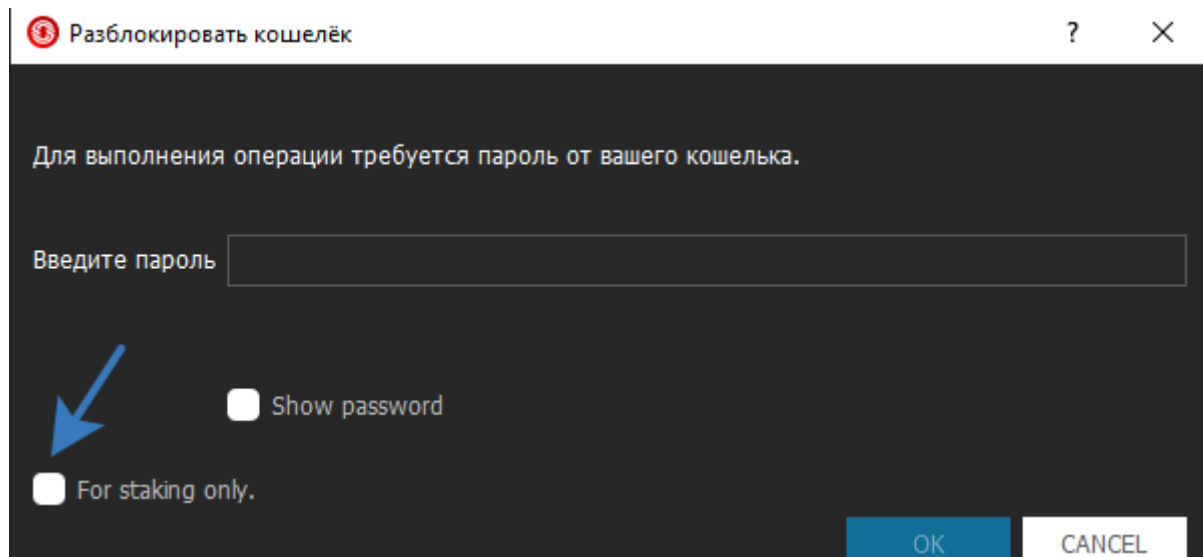
Мы получим какие то данные, которые затем можно перегнать в строку. Ниже показан скриншот для `sendtocontract`

[illegible]

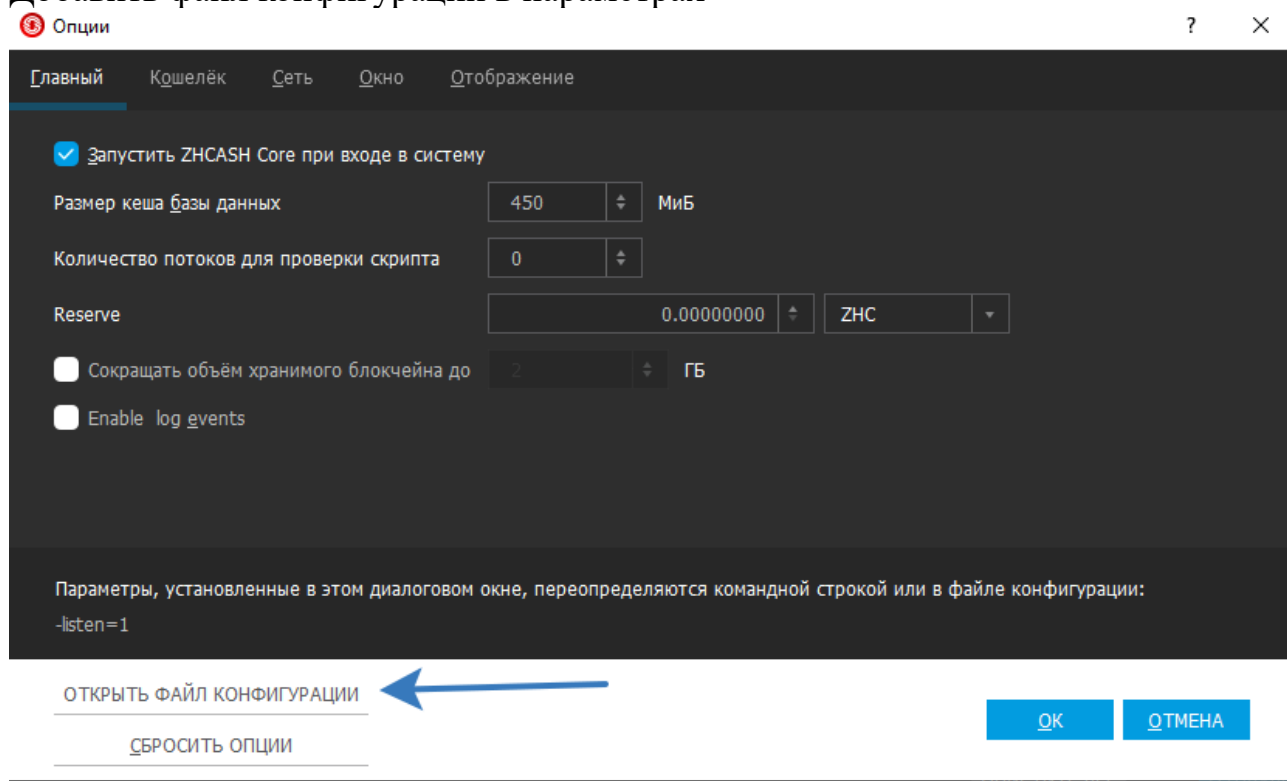
Для того, чтобы взаимодействовать через командную строку `cmd` нужно скачать консольную (серверную) версию кошелька



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=999999999
#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
#rpcallowip=17.2.7.12
#rpcallowip=17.2.7.11
rpcallowip=127.0.0.1
rpcuser=zerohour-rpcuser
rpcpassword=1123581321
```

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 airdrop токенов, отправки транзакций, создание миллиона тестовых токенов, спама, хакинга и тому подобного. Ниже показано как можно вызвать функцию callcontract через cmd

[illegible]

Тестирование смарта

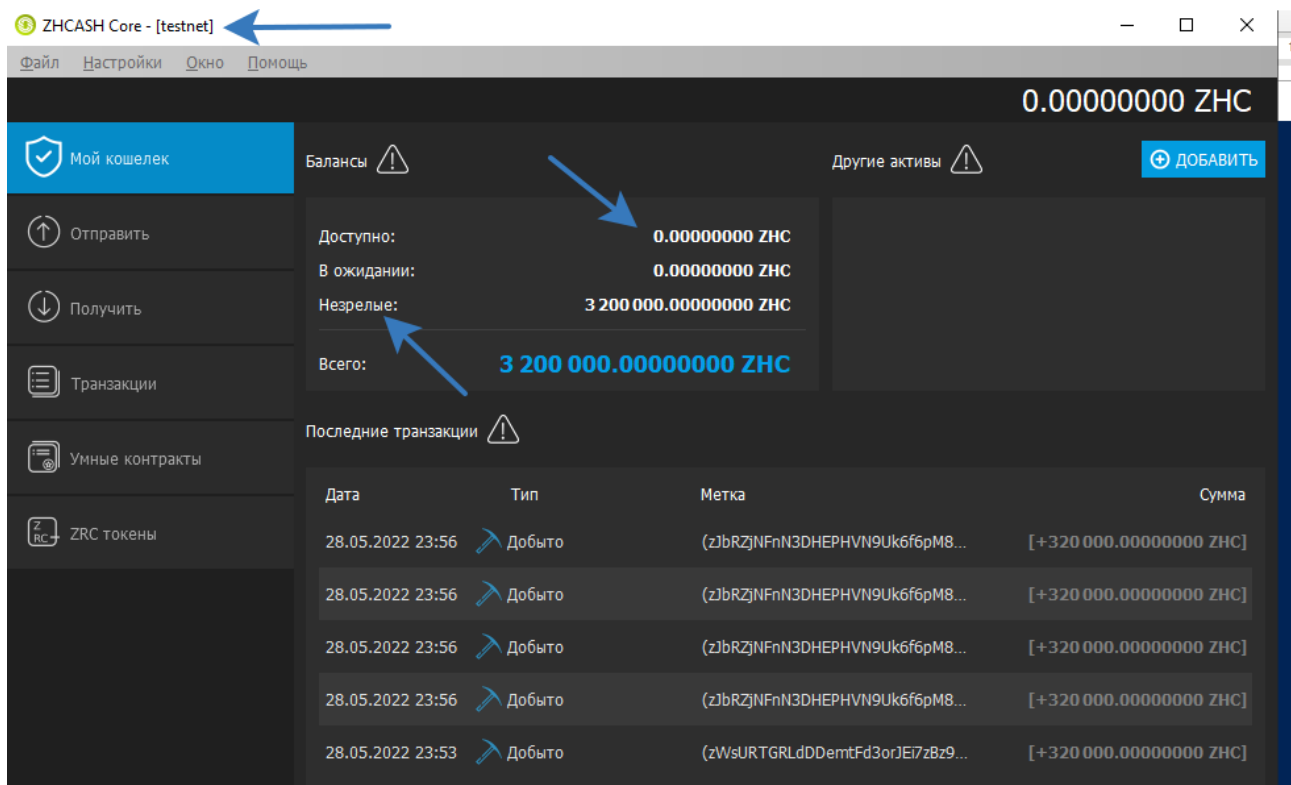
Для тестирования смарта можно использовать три подхода.

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(



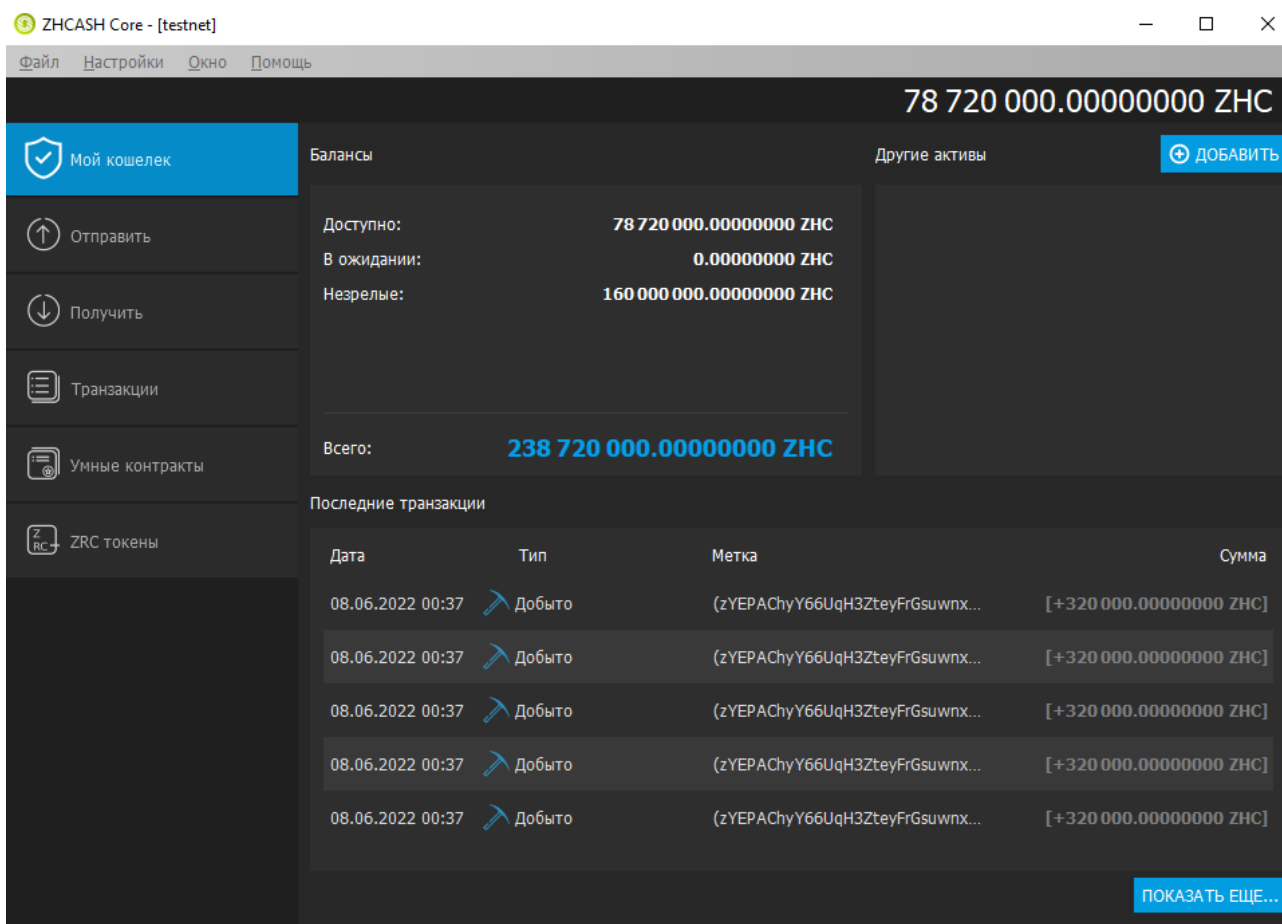
Заходим в консоль, пишем generate 10 и получаем ошибку. Для того, чтобы нагенерить блоки надо запускать с ключами -testnet -deprecatedrpc=generate. Повторяем ещё раз.

```
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.

```
> generate 10|
```

После многократного изнасилования клавиши вверх (для повторения прошлой командны) 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 ± 20 тестировочных монет. При скачивании кошелька сразу доступен отдельный кошелек testnet.

	Дата	Тип	Размер
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

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.

Автором был написан смарт для токена LIFT

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

Автор выражает благодарность инициаторам создания токена LIFT

https://t.me/lift_club с адресом f180d0a911d09853685764a9ad6d366398c50656

Николаю, Арджуну и Денису.

Главному инженеру блокчейна zh Роману, программисту Alex 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.