# Guide

## Preparation of Achain wallets

Introduction of Achain wallets under windows operating system：

1. Official download address：<https://www.achain.com/home.html>
2. Start the wallet（2 kinds of startup method：QT and command line）
3. QT method, double click the icon.
4. Command line method：First go to the directory of the Achain wallet. Then start the wallet using command：

Achain-c.exe --rpcuser admin --rpcpassword 123456 --httpdendpoint 127.0.0.1:8299 --server --data --dir d:\config

Introduction of Achain wallet under Linux operation system：

1. Official download address：<https://github.com/Achain-Dev/Achain_linux>
2. The code need to be compiled. Compiling method refer to document：

<https://github.com/Achain-Dev/Achain_linux/blob/master/linux_installation_guide>

1. After compiling, start the wallet using command：

Achain --rpcuser admin --rpcpassword 123456 --httpdendpoint 127.0.0.1:8299 --server --data-dir c:\user\_data\config

Introduction of the Docker wallet:

1. Official download address：<https://github.com/Achain-Dev/Docker>

2、 To start the wallet please refer to document：<https://github.com/Achain-Dev/Docker/blob/master/README.md>

Introduction of relevant Parameters：

1、 rpcuser : username to start the rpc command;

2、 rpcpassword :password to start the rpc command;

3、 httpdentpoint :ip address of the http request. (Docker wallet don’t need to set this parameter）

4、 data-dir: storage path of the sync data.

### Wallet synchronization

After starting the wallet, the wallet automatically connects the potential node and starts to update block information.

Do not use RPC command until the wallet synchronized to the newest block.

You can use info command to check the synchronization on.

Mainly concern three parameters：

The first is blockchain\_head\_block\_num, this is the current block number。

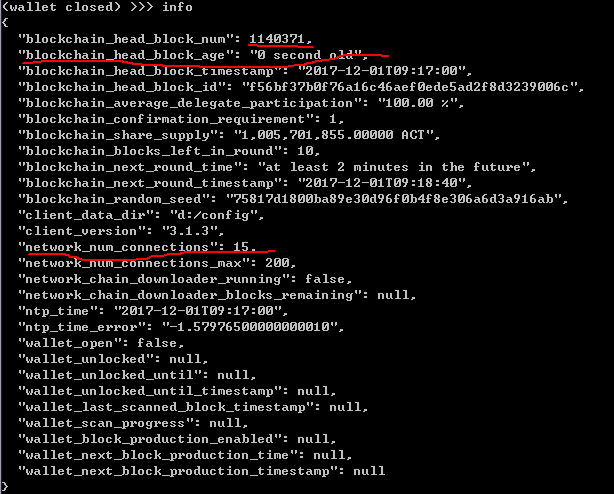
The second is blockchain\_head\_block\_age, this is the age of the current block。We generate new block every10 seconds. Therefore if this parameter is bigger than 10 seconds, the synchronization is not over yet.

The third is network\_num\_connection, this is the current connection number. If this equals 0, the synchronization won’t start. This problem need to be solved in time.

Note: During synchronization, there will be delays or errors when calling RPC command.

Please call RPC command after synchronization.

For example:



### Create wallet and accounts

Step 1 : Use command wallet\_create to create wallet. User accounts are saved in this wallet.

There are two parameters: first is the wallet name. second is the wallet password.

After the creation of the wallet, the wallet is unlock automatically.

For example:

C:\Users\fanyukun\AppData\Local\Temp\1513651597(1).png

Step 2: Create accounts.

There two ways for current exchange to manage users:

First: Create accounts for every user and manage these accounts under wallet. This need to collect funds on your own.

Note: In this way you need to turn off automatic backup or the logs grow too fast.

Command: wallet\_set\_automatic\_backups false

Second: Only create a main account. You can use sub\_address to separate user account by stitching 32bit sub\_address after the main account. (you can choose UUID by removing “-“ ) Benefit: When transferring ACT, the sub\_address will do bookkeeping work. You don’t need to collect your funds.

The sub\_address method is recommended.

RPC command：wallet\_account\_create, the parameter is the account name.

This command return the address of the account. If the return value is none or error, the account name might be registered or invalid.

After doing the work we finish the preparation of the wallet.

C:\Users\fanyukun\AppData\Local\Temp\1513651780(1).png

## Get transaction（block scanning method）

1. Get the chain block.

Call RPC method blockchain\_get\_block\_count to get the header block number.

Use RPC method blockchain\_get\_block to query every block. The parameter is the block number.

Note: The field "user\_transaction\_ids": []. If this field is empty, there is no transaction on this block. You can scan the next block If not empty, then do the transaction query job. user\_transaction\_ids records the trx\_id of transaction.

For example:



Note: trx\_id is from the block fields user\_transaction\_ids

Explaining：ori\_trx\_id and result\_trx\_id only occurs on contract calling. They are not occurred in ACT transaction.

(Transaction id on achain browser: trx\_id for ACT transaction, ori\_trx\_id for contract calling).

ori\_trx\_id records the origin transaction, that is, the transaction relative to ACT. Therefore, when calling contract, ori\_trx\_id only records the costs of ACT, fee and call contract cost limit. As for the contract user are calling, the ori\_trx\_id only records the calling method and parameters. It won’t record the result of the contract. In addition, the result of transaction called by RPC command is ori\_trx\_id.

result\_trx\_id records the complete transaction on the block. Except the ACT transaction we described above, the result\_trx\_id also records whether the calling contract is successful and the result of the calling contracts. The transaction id gotten by scanning block is trx\_id.

2、query ACT transaction

Use trx\_id in field user\_transaction\_ids as parameter.

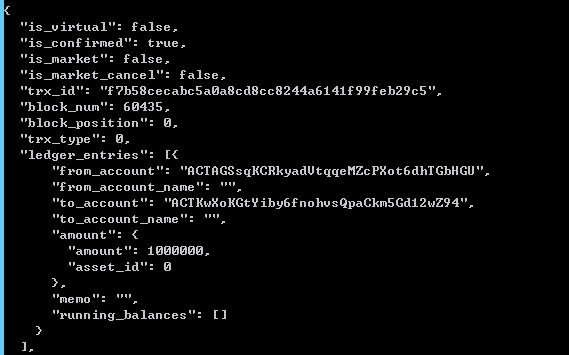
Call RPC command: blockchain\_get\_pretty\_transaction to get the from and to account of this transaction.

from\_account: withdraw address

to\_account : deposit address

amount: 100,000 times of the transaction amount.

For example:



Then call RPC command blockchain\_get\_transaction to get if there is sub-address transaction.

The field alp\_account is the deposit address. The figure is a transaction with sub-address. You can get sub-address from field alp\_account.

For example:



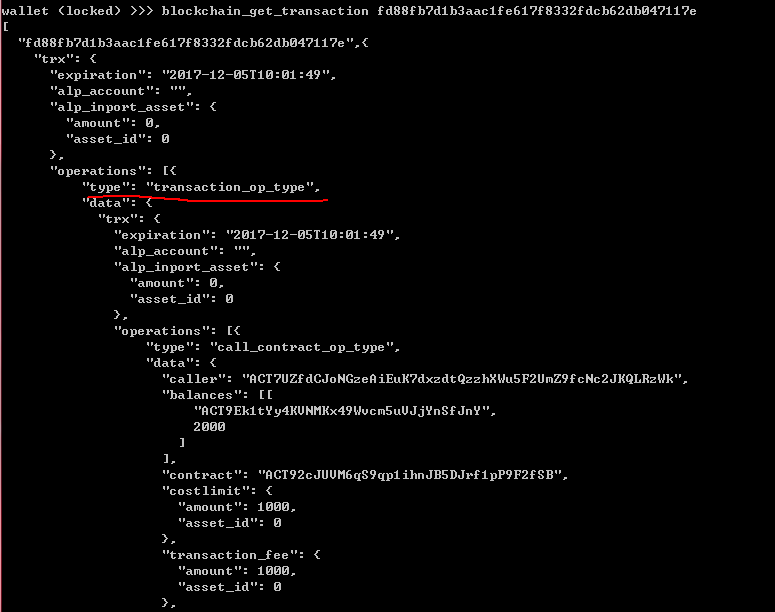
1. **Query contract transaction**

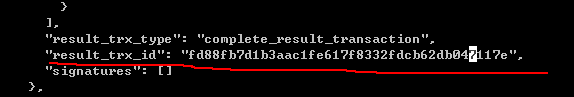
Use trx\_id in field user\_transaction\_ids as parameter.

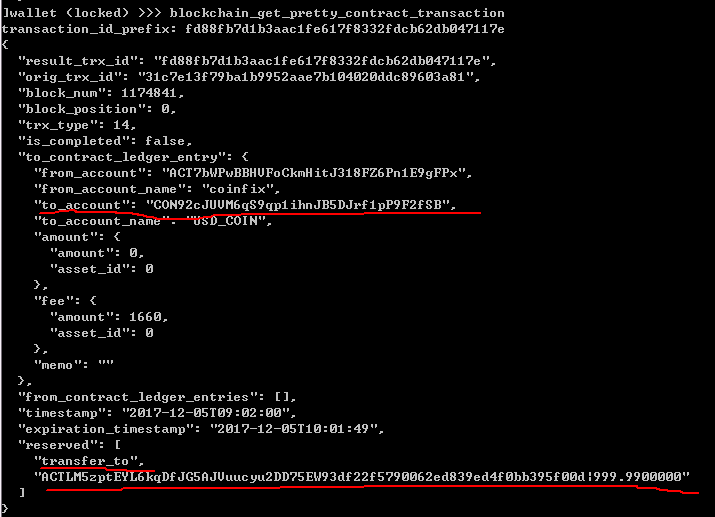
Call RPC command: blockchain\_get\_transaction to get the from and to account of this transaction.

In field operation, if the value of field type is transaction\_op\_type, this is contract transaction. Then get the value of field result\_trx\_id. Note that this result\_trx\_id is at trx layer. Don’t get the wrong field.

Next, use RPC command blockchain\_get\_pretty\_contract\_transaction, the parameter is result\_trx\_id we get previously. We first check the field to\_account whether it is the right token contract id. If not, just pass it. Then we check field reserved. First parameter is the method the contract called. If the value is transfer\_to, this is a contract transaction. If not, skip it. The second parameter is the parameter of method transfer\_to, using to separate different parameters. The first is deposit address, the second is transaction amount. To monitor whether your address has received the contract token, you can check if this address is your own address.







3、Query the balance of contract token

Use RPC command blockchain\_get\_events, the parameter is the block number and result\_trx\_id.

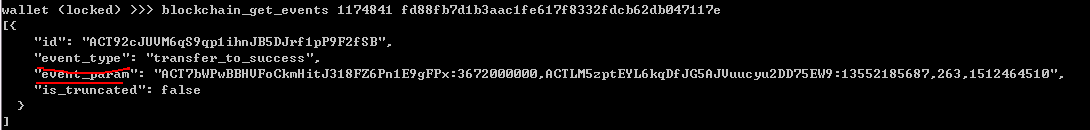
To get event\_type and event\_param.

Among these parameters, event\_type indicate whether the transaction succeed.

If the result is transfer\_to\_success, the transaction succeed. If not, the transaction failed.

There are 4 parameters in field event\_param, separate by “,”. First is withdraw address and balance, separate by ”:”. Second is deposit address and balance. Third is the version (increment). Forth is timestamp.

Note: balance is the balance after this transaction.



4、Query ACT balance by ACT address

Use RPC command: blockchain\_list\_address\_balances. The parameter is ACT address.

Return: The sum of every field balance is the balance of ACT address.

5、Withdraw of contract token

The transfer function of contract token is realized by contract calling.

Therefore we should use RPC command call\_contract to transfer contract token.

call\_contract method needs 6 parameters:

First is contract\_id, start with “CON”.

Second is user account who calls contract, that is, withdraw account.

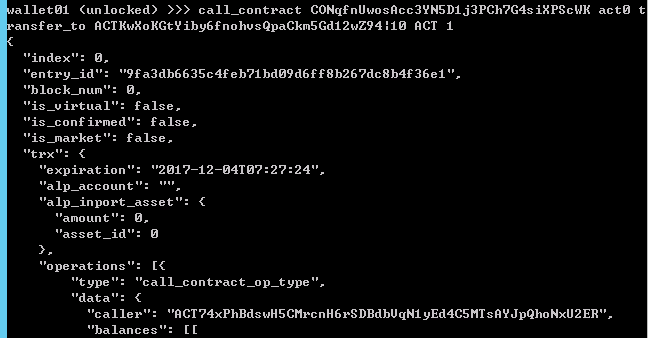
Third is calling method. Here we use method transfer\_to.

Forth is parameter that calling method used. The format is to\_address|amount. The parameters are separate by ”|”.

Fifth parameter is const string “ACT”.

Sixth parameter is the cost limit of contract call. This can be set to any number bigger than 0.01.

Note: Transfer amount up to 5 decimal places. If decimal more than 5, the last will be ignored.



**call\_contract method**

After call\_contract, the block will return the specific information of the transaction.

Here we get field entry\_id，that is ori\_trx\_id we described before.

Call RPC method blockchain\_get\_contract\_result，use entry\_id as parameter.

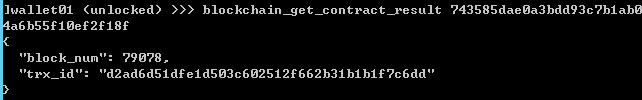
There are two parameters in the result. The first is the block number that the transaction occurred. The second is result\_trx\_id. Then use these two parameter to call RPC method blockchain\_get\_events.

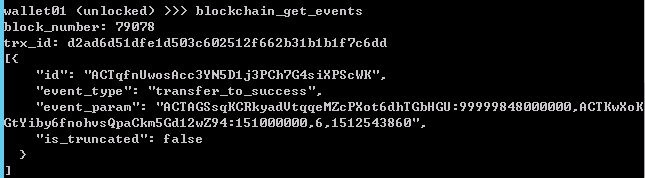
Then we will get event\_type and event\_param.

Among these parameters, event\_type indicate whether the transaction succeed.

If the result is transfer\_to\_success, the transaction succeed. If not, the transaction failed.

There are 4 parameters in field event\_param, separate by “,”. First is withdraw address and balance, separate by ”:”. Second is deposit address and balance. Third is the version (increment). Forth is timestamp.

Note: after call\_contract, the transaction is not record in the block immediately. Therefore when querying transaction information, we should wait until the new block generates or the query result may be empty. We generate new block every10 seconds 



6、Withdraw ACT

Call RPC method wallet\_transfer\_to\_address to withdraw ACT. Before calling this method, we should open and unlock wallet. This method need 4 parameters. The first is withdraw ACT amount. This is true transaction amount. The second is const string “ACT”. The third is withdraw account name. The forth is the deposit ACT address. This method will return a transaction id, which is trx\_id.