CDS Node Data Summary

Internal File：V.0.0.1

Issue Time：2019-11-08

**Content**

* Description of file version 3
* References 3
* Manual purpose 3
* Statement 3
* Terms definition and abbreviations 3

1. Download and compile 4

1.1 Open source address 4

1.2 Compile 4

1.2.1 Install JDK 4

1.2.2 Install the compilation tool Maven3 4

1.2.3 Compile 4

1. Start node 5
   1. Profile 5

2.1.1、Start ports and synchronization nodes 5

2.1.2、Node data storage path 5

2.1.3、rpc function 5

2.2、Start node 5

3 Node interface 6

3.1 Use web3 link nodes 6

3.2 Check address balance 6

3.3 Get the transaction nonce 7

3.4 Create node wallet 7

3.5 Get node wallet address 7

3.6 Use node wallet to send transaction 7

3.7 Use external wallet to send transaction 8

4 Something worth noting 9

4.1 Unable to load dll dynamic link library using windows server 9

4.2 Using windows server, found that the P2P network is unstable 9

4.3 Blockchain browser 9

4.4 About transaction confirmation 9

4.5 About smart contract 9

* Description of file version

Table 1 Version description

|  |  |  |  |
| --- | --- | --- | --- |
| Version | Release time | Revised chapter | Author |
| V.0.0.1 | 2019-11-08 | Build summary materials | Prings |
|  |  |  |  |
|  |  |  |  |

* References

[1] https://jingyan.baidu.com/article/e8cdb32b2699cb37042bad59.html

[2] https://blog.csdn.net/williamyi96/article/details/78268595

[3] https://www.cnblogs.com/xixihuang/p/5569203.html

* Manual purpose

Summary of node information, used for technical description and problem solving in the process of listing

* Statement

Free to use and spread

* Terms definition and abbreviations

Table 2 Terms definition and abbreviations

| Number | Abbreviation | Description |
| --- | --- | --- |
|  |  |  |
|  |  |  |
|  |  |  |

# Download and compile

## 1.1 Open source address

no data

## 1.2 Compile

### 1.2.1 Install JDK

The subject of the node is developed in java language. Therefore, you need to ensure that Java 8 is installed correctly (or above, both OpenJdk or oracle Jdk have passed the test). Please refer to the following installation details:

Windows: https://jingyan.baidu.com/article/e8cdb32b2699cb37042bad59.html

linux(Take ubantu as an example): https://blog.csdn.net/williamyi96/article/details/78268595

### 1.2.2 Install the compilation tool Maven3

The tool for compiling the code is Maven3

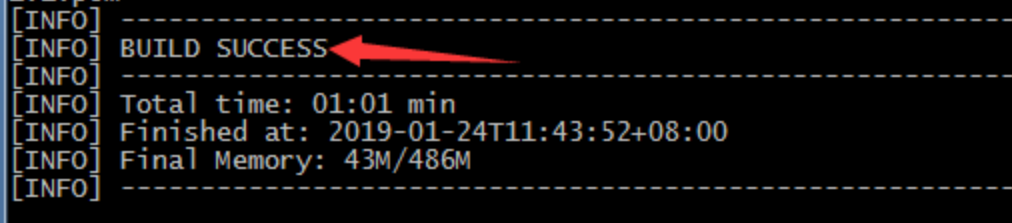
Please refer to the installation details:https://www.cnblogs.com/xixihuang/p/5569203.html

### 1.2.3 Compile

Download the source code from githup, unzip, and enter the code directory cds-main-blockchain-node

Enter the installation directory under windows cmd or linux terminal

Execute the MVN install command to compile the code. After the compilation is successful, it appears as shown in the figure



The target folder will appear in the directory after compilation and execution

Enter this folder and you can see cds-main-blockchain-node-0.0.1.jar. That is, the product package

# Start node

## Profile

In the code directory, there is an external configuration file blockchain.conf, which is used to control the behavior of nodes, as follows:

### 2.1.1、Start ports and synchronization nodes

peer {

listen.port = 30777 #Start port

active = [ #Active node. More nodes means faster synchronization

{

ip = ""

port = 30777

nodeId = ""

}

]

trusted = [ #Trust node. More nodes means faster synchronization

{

ip = ""

nodeId = ""

}

]

}

### 2.1.2、Node data storage path

database {

dir = "./database" #data storage path

}

### 2.1.3、rpc function

modules {

rpc {

enabled = true #Whether to enable rpc function

port = 8866

ipWhiteList=["192.168.\*.\*","47.75.96.\*"] #White List

}

}

## 2.2、Start node

Command to start node:

Windows:

java -jar -Xms1024m -Xmx4096m cds-blockchain-node-0.0.1.jar

linux(Take ubantu as an example):

nohup java -jar -Xms1024m -Xmx4096m cds-blockchain-node-0.0.1.jar >/dev/null 2>&1 &

Note: In the directory where the product package is located, create a config folder to place configuration filesblockchain.conf

# 3 Node interface

The node inherited Ethereum and implemented most of the web3 interfaces (of course the web3j you are using is also compatible). Before calling web3, please make sure that the external configuration user.conf has started the json rpc module and the IP whitelist has been correctly configured.

All the following sample codes use web3.js (the same principle as web3J), and the nodejs dependency configuration is as follows:

{

"name": "web3 test",

"private": true,

"version": "0.0.1",

"engines": {

"node": ">=5.0.0"

},

"devDependencies": {

"web3": "^1.0.0-beta.34",

"solc": "^0.4.22",

"ethereumjs-tx": "^1.3.7"

},

}

## 3.1 Use web3 link nodes

The current way of link web3 supports http, and the node code is as follows

// Load web3 module

var Web3 = require('web3');

// Create web3 link

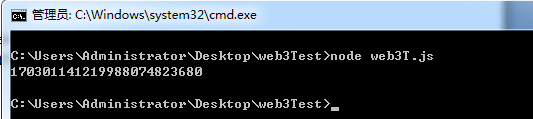
var web3 = new Web3(new Web3.providers.HttpProvider("http://127.0.0.1:8866"));

## 3.2 Check address balance

var Web3 = require('web3');

var web3 = new Web3(new Web3.providers.HttpProvider("http://127.0.0.1:8866"));

web3.eth.getBalance("0x55b6f781cc8dd985953ef36a7dce36ea4a09109f").then(console.log);



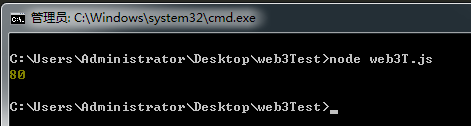
Balance is the same as Ethereum, 18-bit precision, and can be used in web3 toWei and fromWei methods

## 3.3 Get the transaction nonce

var Web3 = require('web3');

var web3 = new Web3(new Web3.providers.HttpProvider("http://127.0.0.1:8866"));

web3.eth.getTransactionCount("0x55b6f781cc8dd985953ef36a7dce36ea4a09109f","pending").then(console.log);

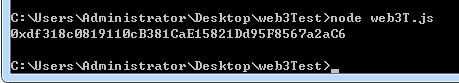


## 3.4 Create node wallet

var Web3 = require('web3');

var web3 = new Web3(new Web3.providers.HttpProvider("http://127.0.0.1:8866"));

web3.eth.personal.newAccount('!@superpassword').then(console.log);



## 3.5 Get node wallet address

var Web3 = require('web3');

var web3 = new Web3(new Web3.providers.HttpProvider("http://127.0.0.1:8866"));

web3.eth.getAccounts().then(console.log);

图片 5

## 3.6 Use node wallet to send transaction

var Web3 = require('web3');

var web3 = new Web3(new Web3.providers.HttpProvider("http://127.0.0.1:8866"));

var fromAddress = "0x543b6e1125c84810c9f714f81147b553Eb2B2977";//Node wallet address

var receiverAccount = "0xdf318c0819110cB381CaE15821Dd95F8567a2aC6";

//Unlock the node wallet

web3.eth.personal.unlockAccount(fromAddress, "test").then(openAccountState=>{

if(openAccountState){

web3.eth.sendTransaction({

from: fromAddress,

to: receiverAccount,

value: web3.utils.toWei("0.1", "ether"),

gasLimit: "21000",

gasPrice: web3.utils.toWei("15", "gwei")

}).then(function(receipt){

console.log(receipt);

});

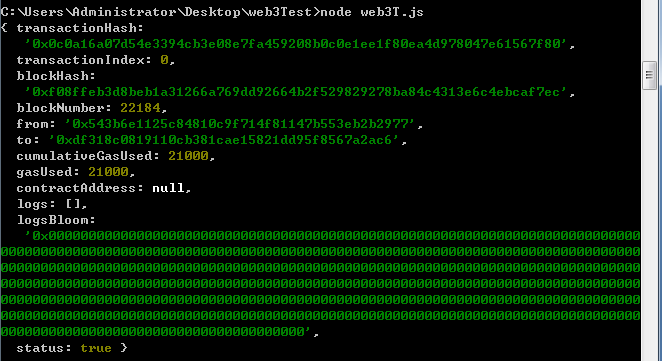
}else{

console.log("Fail to unlock node account");

}

});

It is worth noting that the cds node has replaced the mining algorithm, so the block speed is about 2 minutes per block, which means that the time to obtain the transaction receipt is about 2 minutes. So it will take up to 2 minutes to run this code and the following output will appear. Please be patient



## 3.7 Use external wallet to send transaction

For some applications with higher security levels, wallets are generally not handed over to nodes for management. You can use any Ethereum wallet generation tool and signature tool to generate wallet address and signature transactions. But regardless of the method, in the end, you need to use web3.eth.sendSignedTransaction to send the transaction. The sample code is as follows:var Web3 = require('web3');

var web3 = new Web3(new Web3.providers.HttpProvider("http://127.0.0.1:8866"));

var Tx = require('ethereumjs-tx');

//Address key

var privateKey = Buffer.from('a6b6f705ae5bb91c3ad1049995490c028d7c1d64f116e740a9fdc5bd49c75c1b', 'hex');

//Send address

var fromAddress = "0x578174A9303c5D48Ca7cB2450033Ef08C450dCCE";

//Receive address

var receiverAccount = "0xdf318c0819110cB381CaE15821Dd95F8567a2aC6";

var rawTx = {

from: fromAddress,

to: receiverAccount,

value: web3.utils.toHex(web3.utils.toWei("0.01", "ether")),

gasLimit: web3.utils.toHex(21000),

gasPrice: web3.utils.toHex(web3.utils.toWei("15", "gwei"))

}

var tx = new Tx(rawTx);

tx.sign(privateKey);

var serializedTx = tx.serialize();

web3.eth.sendSignedTransaction('0x' + serializedTx.toString('hex'),function(err, hash){

if (!err) {

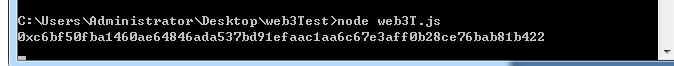
console.log(hash);

} else {

console.error(err);

}

});



# 4 Something worth noting

## 4.1 Unable to load dll dynamic link library using windows server

If your server is windows, the node needs some additional dynamic link libraries, generally you only need to install Visual Studio 2017.

## 4.2 Using windows server, found that the P2P network is unstable

If there is a node reconnection during the process, please pay attention to whether the server's time stamp is correct. There are many ways to synchronize time stamps in windows, please refer to it yourself.

## 4.3 Blockchain browser

http://47.75.96.91/

## 4.4 About transaction confirmation

In order to avoid rolling back transactions due to block forks, we recommend that you need to perform a second confirmation of the transaction in the node. According to the current transaction volume, the block time for the second confirmation can be the first 6 to 8 blocks of the current latest block. If the node has generated the transaction receipt information and the status is successful, it can be considered that the transaction has been successful.

## 4.5 About smart contract

The smart contract was compiled and deployed successfully under solc4.25, and the EVM bytecode was verified internally.