Wiki文章题目：初链API

作者：lesterli

**TrueChain API**

**API Introduction**

The TrueChain supported APIs, main module: admin, miner, personal, etc. Those APIs are provided using JSON-RPC.

Getrue comes with JavaScript console and it supports all describe APIs here. To provide these APIs through Getrue RPC, using the --${interface}api command-line arguments to use it. The ${interface} can be HTTP rpc, WebSocket ws and Unix socket ipc, or Windows pipe.

For example: getrue --ipcapi admin,etrue,miner --rpcapi etrue,web3 --rpc

* Enable administrator, official DApp and miner API with the IPC interface;
* Enable official DApp and web3 API via HTTP interface;

The HTTP RPC interface must be explicitly enabled using the --rpc label.

NOTE: The APIs provided through HTTP(rpc) or WebSocket (ws) interfaces will open for everyone to access(DApps, browser tabs, etc). By default, Getrue enable all APIs through IPC(ipc), and enable etrue, net and web3 APIs through HTTP and WebSocket interface.

To determine which APIs the interface provides, call the JSON-RPC method modules.

For example, through the ipc interface invoke JSON-RPC on Unix:

echo '{"jsonrpc":"2.0","method":"rpc\_modules","params":[],"id":1}' | nc -U $datadir/getrue.ipc

It will list all enabled modules, and including the version number:

{

"id":1,

"jsonrpc":"2.0",

"result":{

"admin":"1.0",

"debug":"1.0",

"etrue":"1.0",

"miner":"1.0",

"net":"1.0",

"personal":"1.0",

"rpc":"1.0",

"txpool":"1.0",

"web3":"1.0"

}

}

**API List**

The commonly APIs include: admin, miner, personal, etc, as showed in the following:

* admin: node management
* miner: mine management
* personal: account management

**Admin**

Using the admin API, you can access multiple RPC methods that will allow to control the Getrue instance, including but not limited to peer and RPC management.

**admin\_addPeer**

The addPeer method will add the new remote node to the tracked node list. The node will always try to keep connections with these nodes, and reconnect if the remote connection is disconnected.

The method accepts a argument. The parameter is the enode URL of the remote peer, and returns BOOL to indicating whether peer accepts the connection or some error has occurred.

Console call:

admin.addPeer(url)

RPC invokes:

{"method": "admin\_addPeer", "params": [url]}

Example:

> admin.addPeer("enode://a979fb575495b8d6db44f750317d0f4622bf4c2aa3365d6af7c284339968eef29b69ad0dce72a4d8db5ebb4968de0e3bec910127f134779fbcb0cb6d3331163c@52.16.188.185:30303")

true

**admin\_datadir**

The datadir method be used to query Getrue node store data directory.

Console call:

admin.datadir

RPC invokes:

{"method": "admin\_datadir"}

Example:

> admin.datadir

"/home/true/truedata"

**admin\_nodeInfo**

The nodeInfo method be used to get Getrue node information, such as the enode URL of the node.

Console call:

admin.nodeInfo

RPC invokes:

{"method": "admin\_nodeInfo"}

Example:

> admin.nodeInfo

{

enode: "enode://44826a5d6a55f88a18298bca4773fca5749cdc3a5c9f308aa7d810e9b31123f3e7c5fba0b1d70aac5308426f47df2a128a6747040a3815cc7dd7167d03be320d@[::]:30303",

id: "44826a5d6a55f88a18298bca4773fca5749cdc3a5c9f308aa7d810e9b31123f3e7c5fba0b1d70aac5308426f47df2a128a6747040a3815cc7dd7167d03be320d",

ip: "::",

listenAddr: "[::]:30303",

name: "Getrue/v0.7.2-unstable-46f28475/linux-amd64/go1.10.1",

ports: {

discovery: 30303,

listener: 30303

},

protocols: {

etrue: {

difficulty: null,

genesis: "0xd4e56740f876aef8c010b86a40d5f56745a118d0906a34e69aec8c0db1cb8fa3",

head: "0xb83f73fbe6220c111136aefd27b160bf4a34085c65ba89f24246b3162257c36a",

network: 20

}

}

}

**admin\_peers**

The peer method be used to get all remote nodes connected information.

Console call:

admin.peers

RPC invokes:

{"method": "admin\_peers"}

Example:

> admin.peers

[{

caps: ["eth/61", "eth/62", "eth/63"],

id: "08a6b39263470c78d3e4f58e3c997cd2e7af623afce64656cfc56480babcea7a9138f3d09d7b9879344c2d2e457679e3655d4b56eaff5fd4fd7f147bdb045124",

name: "Getrue/v0.7.2-unstable-46f28475/linux-amd64/go1.10.1",

network: {

localAddress: "192.168.0.104:51068",

remoteAddress: "71.62.31.72:30303"

},

protocols: {

etrue: {

difficulty: null,

head: "5794b768dae6c6ee5366e6ca7662bdff2882576e09609bf778633e470e0e7852",

version: 63

}

}

}, /\* ... \*/ {

caps: ["eth/61", "eth/62", "eth/63"],

id: "fcad9f6d3faf89a0908a11ddae9d4be3a1039108263b06c96171eb3b0f3ba85a7095a03bb65198c35a04829032d198759edfca9b63a8b69dc47a205d94fce7cc",

name: "Getrue/v0.7.2-unstable-46f28475/linux-amd64/go1.10.1",

network: {

localAddress: "192.168.0.104:55968",

remoteAddress: "121.196.232.205:30303"

},

protocols: {

etrue: {

difficulty: null,

head: "5794b768dae6c6ee5366e6ca7662bdff2882576e09609bf778633e470e0e7852",

version: 63

}

}

}]

**admin\_startRPC**

The startRPC method be used to starts HTTP-based JSON RPC, and handle client Web requests. Follow parameter are optional:

* host: Network address for listening(default: localhost)
* port: Network port for listening(default:8545)
* cors: cross-domain access(default: "")
* apis: APIs module(default:"etrue,net,web3")

Console call:

admin.startRPC(host, port, cors, apis)

RPC invokes:

{"method": "admin\_startRPC", "params": [host, port, cors, apis]}

Example:

> admin.startRPC("127.0.0.1", 8545)

true

**admin\_stopRPC**

The stopRPC method be used to closes the opened HTTP RPC.

Console call:

admin.stopRPC()

RPC invokes:

{"method": "admin\_stopRPC"

Example:

> admin.stopRPC()

true

**admin\_stopWS**

The stopWS method be used to closes the WebSocket RPC.

Console call:

admin.stopWS()

RPC invokes:

{"method": "admin\_stopWS"

Example:

> admin.stopWS()

true

**Miner**

The miner APIs be used to management the mining operation and set various configuration.

**miner\_setGasPrice**

Set the minimum acceptable Gas price for mining transactions. Any trade below this limit will not be accepted.

Console call:

miner.setGasPrice(number)

RPC invokes:

{"method": "miner\_setGasPrice", "params": [number]}

**miner\_start**

Start the CPU mining with given threads number.

Console call:

miner.start(number)

RPC invokes:

{"method": "miner\_start", "params": [number]}

**miner\_stop**

Stop CPU mining.

Console call:

miner.stop()

RPC invokes:

{"method": "miner\_stop", "params": []}

**miner\_setEtherBase**

Set the reward address.

Console call:

miner.setEtherbase(address)

RPC invokes:

{"method": "miner\_setEtherbase", "params": [address]}

**Personal**

The personal API manages the private keys in the keystore.

**personal\_listAccounts**

The listAccounts method will return all TrueChain accounts in the keystore.

Console call:

personal.listAccounts

RPC invokes:

{"method": "personal\_listAccounts", "params": []}

Example:

> personal.listAccounts

["0x5e97870f263700f46aa00d967821199b9bc5a120", "0x3d80b31a78c30fc628f20b2c89d7ddbf6e53cedc"]

**personal\_lockAccount**

Remove the private key with the given address from memory. This account cannot send trade.

Console call:

personal.lockAccount(address)

RPC invokes:

{"method": "personal\_lockAccount", "params": [string]}

**personal\_newAccount**

Generate the new private key and store it in the keystore. The key file is encrypted with the given password, and returns the address of the new account.

Console call:

personal.newAccount()

RPC invokes:

{"method": "personal\_newAccount", "params": [string]}

Example:

> personal.newAccount()

Passphrase:

Repeat passphrase:

"0x5e97870f263700f46aa00d967821199b9bc5a120"

The password can also be provided as argument.

Example:

> personal.newAccount("h4ck3r")

"0x3d80b31a78c30fc628f20b2c89d7ddbf6e53cedc"

**personal\_unlockAccount**

Unlock the given address.

in JavaScript console, both password and lock duration are optional.

Unencrypted keys will remain in memory until the lock duration expires.

The account can be used with eth\_sign and eth\_sendTransaction when unlocked.

Console call:

personal.unlockAccount(address, passphrase, duration)

RPC invokes:

{"method": "personal\_unlockAccount", "params": [string, string, number]}

Example:

> personal.unlockAccount("0x5e97870f263700f46aa00d967821199b9bc5a120")

Unlock account 0x5e97870f263700f46aa00d967821199b9bc5a120

Passphrase:

true

The password and lock duration can be provided as argument.

> personal.unlockAccount("0x5e97870f263700f46aa00d967821199b9bc5a120", "foo", 30)

true

If you want enter the password and override the default unlock duration, pass null as the password.

> personal.unlockAccount("0x5e97870f263700f46aa00d967821199b9bc5a120", null, 30)

Unlock account 0x5e97870f263700f46aa00d967821199b9bc5a120

Passphrase:

true

**personal\_sendTransaction**

Verify the given password and submit the transaction.

The parameter same with eth\_sendTransaction and contains from address. If the password can be use decrypt tx.form's, the transaction is validated, the signature is send to the network. The account cannot be used in other RPC calls if it locked in the node.

Console call:

personal.sendTransaction(tx, passphrase)

RPC invokes:

{"method": "personal\_sendTransaction", "params": [tx, string]}

Example:

> var tx = {from: "0x391694e7e0b0cce554cb130d723a9d27458f9298", to: "0xafa3f8684e54059998bc3a7b0d2b0da075154d66", value: web3.toWei(1.23, "ether")}

undefined

> personal.sendTransaction(tx, "passphrase")

0x8474441674cdd47b35b875fd1a530b800b51a5264b9975fb21129eeb8c18582f