

**井通科技**

**jingtum-lib接口说明**

V2.1.6

**版本历史**

|  |  |  |  |
| --- | --- | --- | --- |
| **版本** | **简介** | **作者** | **日期** |
| 1.0.0 | 初版主要接口说明 | 吴丹 | 2018/2/28 |
| 2.0.0 | 整体接口说明 | 吴丹 | 2018/3/2 |
| 2.0.1 | Remote类增加返回结果说明 | 吴丹 | 2018/3/15 |
| 2.0.2 | Remote类增加合约方法(Lua版) | 吴丹 | 2018/5/31 |
| 2.0.3 | Remote类增加设置和获取挂单佣金的方法；挂单可选传入自己的应用来源标记；utils.processTx()方法过滤交易记录时，成交的effect中增加费率，且got是去除挂单手续费的结果； | 吴丹 | 2018/8/15 |
| 2.1.0 | 增加合约类接口(solidity版)，文档最后附solidity ERC20源码； | 吴丹 | 2019/3/5 |
| 2.1.1 | 文档最后增加solidity ERC721源码； | 吴丹 | 2019/5/8 |
| 2.1.2 | solidity调用合约新增amount字段;删除 ERC721源码(版本老); | 吴丹 | 2019/5/14 |
| 2.1.3 | [utils.processTx方法返回结果增加挂单价格及多方撮合数量、价格](http://git.jingtum.com/root/jingtum-lib/commit/407cfcbb1939414b755efaa7fefd1b33bb8b08bf)； | 吴丹 | 2019/5/24 |
| 2.1.4 | 支付amount解决41000变成40999精度不准问题；采用bignumber.js最新版，支持整数+小数20位数字； | 吴丹 | 2019/5/30 |
| 2.1.5 | 新增工具类说明；lua版合约函数名foo改为func | 吴丹 | 2019/06/04 |
| 2.1.6 | 设置挂单佣金接口(buildBrokerageTx)增加收费账号feeAccount字段；获取挂单佣金及获取用户挂单接口返回增加应用来源AppType、收费币种FeeCurrency、手续费分母OfferFeeRateDen、手续费分子OfferFeeRateNum四个字段；processTx方法增加变动币种余额balances、总共收取的挂单手续费totalRate(成交的单子才有此字段)字段；effects中增加应用来源app、挂单费率rate字段； | 吴丹 | 2019/06/12 |

[1 安装 1](#_Toc699)

[2 项目文件结构 1](#_Toc27316)

[3 创建钱包 1](#_Toc25677)

[4 Remote类 2](#_Toc5349)

[4.1 创建Remote对象 3](#_Toc21791)

[4.2 创建连接 4](#_Toc24541)

[4.3 关闭连接 5](#_Toc22801)

[4.4 请求底层服务器信息 6](#_Toc28186)

[4.5 获取最新账本信息 7](#_Toc1602)

[4.6 获取某一账本具体信息 9](#_Toc27763)

[4.7 查询某一交易具体信息 11](#_Toc20628)

[4.8 请求账号信息 14](#_Toc14337)

[4.9 获得账号可接收和发送的货币 16](#_Toc32378)

[4.10 获得账号关系 19](#_Toc19521)

[4.11 获得账号挂单 22](#_Toc30982)

[4.12 获得账号交易列表 25](#_Toc863)

[4.13 获得市场挂单列表 27](#_Toc31116)

[4.14 获得挂单佣金设置信息 30](#_Toc16844)

[4.15 支付 33](#_Toc26978)

[4.15.1 创建支付对象 33](#_Toc12172)

[4.15.2 传入密钥 33](#_Toc11386)

[4.15.3 设置备注 33](#_Toc29916)

[4.15.4 提交支付 34](#_Toc5608)

[4.16 设置关系 36](#_Toc23837)

[4.16.1 创建关系对象 36](#_Toc32156)

[4.16.2 传入密钥 37](#_Toc6711)

[4.16.3 关系设置 37](#_Toc26830)

[4.17 设置账号属性 -----------待完善 40](#_Toc29633)

[4.17.1 创建属性对象 41](#_Toc754)

[4.17.2 传入密钥 41](#_Toc18608)

[4.17.3 属性设置 41](#_Toc21299)

[4.18 挂单 42](#_Toc16233)

[4.18.1 创建挂单对象 42](#_Toc15033)

[4.18.2 传入密钥 43](#_Toc8032)

[4.18.3 提交挂单 43](#_Toc27220)

[4.19 取消挂单 47](#_Toc26796)

[4.19.1 创建取消挂单对象 47](#_Toc12860)

[4.19.2 传入密钥 47](#_Toc8500)

[4.19.3 取消挂单 47](#_Toc25858)

[4.20 部署合约（Lua版） 50](#_Toc19561)

[4.20.1 创建部署合约对象 50](#_Toc3821)

[4.20.2 传入密钥 50](#_Toc27048)

[4.20.3 部署合约 51](#_Toc3412)

[4.21 执行合约（Lua版） 54](#_Toc30382)

[4.21.1 创建执行合约对象 54](#_Toc11700)

[4.21.2 传入密钥 54](#_Toc7889)

[4.21.3 执行合约 54](#_Toc31385)

[4.22 设置挂单佣金 57](#_Toc17624)

[4.22.1 创建挂单佣金对象 57](#_Toc19591)

[4.22.2 传入密钥 58](#_Toc17569)

[4.22.3 设置挂单佣金 58](#_Toc10128)

[4.23 部署合约（Solidity版） 62](#_Toc29436)

[4.23.1 创建合约部署对象 62](#_Toc29243)

[4.23.2 传入密钥 62](#_Toc21863)

[4.23.3 部署合约 62](#_Toc14556)

[4.24 调用合约（Solidity版） 79](#_Toc6278)

[4.24.1 创建合约调用对象 79](#_Toc2202)

[4.24.2 传入密钥 79](#_Toc17301)

[4.24.3 调用合约 80](#_Toc1970)

[4.25 监听事件 88](#_Toc22560)

[5 Request类 89](#_Toc23733)

[5.1 指定账本 89](#_Toc18630)

[5.2 提交请求 90](#_Toc17621)

[6 Transaction类 91](#_Toc18134)

[6.1 获得交易账号 91](#_Toc31100)

[6.2 获得交易类型 92](#_Toc9727)

[6.3 传入私钥 93](#_Toc18372)

[6.4 添加备注 93](#_Toc20087)

[6.5 提交请求 94](#_Toc4235)

[7 工具类 95](#_Toc14645)

[7.1 16进制转字符串 95](#_Toc24113)

[7.2 字符串转16进制 96](#_Toc3902)

[7.3 判断amount是否有效 96](#_Toc25927)

[7.4 判断市场挂单的货币对是否有效 97](#_Toc3088)

[7.5 统一基础货币SWT的amount格式 98](#_Toc12214)

[7.6 判断currency是否合法 98](#_Toc1201)

[7.7 判断hash是否合法 99](#_Toc28202)

[7.8 判断address是否合法 99](#_Toc16367)

[7.9 判断secret是否合法 100](#_Toc2500)

[7.10 从账号角度过滤交易信息 101](#_Toc29130)

[7.11 交易效果effects说明 109](#_Toc8283)

[8 底层常见错误附录 114](#_Toc31532)

[9 Solidity ERC20源码 118](#_Toc10204)

# 安装

npm install --save jingtum-lib

# 项目文件结构

jingtum-lib库基于ws协议跟底层交互，其中ws封装到Server类中，Server类是一个内部类，不对外开放；Server类封装在Remote类中，Remote类提供对外访问接口并可创建两类对象：Get方式请求的Request对象和Post方式请求的Transaction对象，这两类对象都通过submit()方法提交数据到底层。文件结构图如下：

WS通讯协议

Server(内部类)

Remote类

Request对象(GET请求)

Transaction对象(POST请求)

# 创建钱包

首先引入jingtum-lib库的Wallet对象，然后使用以下两种方法创建钱包

方法1：Wallet.generate();

方法2：Wallet.fromSecret(secret);

参数：

|  |  |  |
| --- | --- | --- |
| 参数 | 类型 | 说明 |
| secret | String | 井通钱包私钥 |

例子：

//创建Wallet对象

var jlib = require('jingtum-lib');

var Wallet = jlib.Wallet;

//方式一

var w1 = Wallet.generate();

console.log(w1);

//方式二

var w2 = Wallet.fromSecret('ss2A7yahPhoduQjmG7z9BHu3uReDk');

console.log(w2);

返回的结果信息：

|  |  |  |  |
| --- | --- | --- | --- |
| 参数 | | 类型 | 说明 |
|  | secret | String | 井通钱包私钥 |
| address | String | 井通钱包地址 |

# Remote类

Remote是跟井通底层交互最主要的类，它可以组装交易发送到底层、订阅事件及从底层拉取数据。提供以下方法：

\* Remote(options)

\* connect(callback)

\* disconnect()

\* requestServerInfo()

\* requestLedgerClosed()

\* requestLedger(options)

\* requestTx(options)

\* requestAccountInfo(options)

\* requestAccountTums(options)

\* requestAccountRelations(options)

\* requestAccountOffers(options)

\* requestAccountTx(options)

\* requestOrderBook(options)

\* requestBrokerage(options)

\* buildPaymentTx(options)

\* buildRelationTx(options)

\* buildAccountSetTx(options)

\* buildOfferCreateTx(options)

\* buildOfferCancelTx(options)

\* deployContractTx(options)

\* callContractTx(options)

\* buildBrokerageTx(options)

\* initContract(options)

\* invokeContract(options)

\* on('transactions',callback)

\* on('ledger\_closed',callback)

## 创建Remote对象

方法：new Remote(options);

参数：

|  |  |  |
| --- | --- | --- |
| 参数 | 类型 | 说明 |
| server | String | 井通底层服务地址 |
| local\_sign | Boolean | 交易是否以本地签名的方式发送给底层 |

例子：

var jlib = require('jingtum-lib');

var Remote = jlib.Remote;

var remote = new Remote({server: 'ws://ts5.jingtum.com:5020', local\_sign:true});

## 创建连接

每个Remote对象都应该首先手动连接底层，然后才可以请求底层的数据。请求结果在回调函数callback中。

方法：connect(callback)

参数：回调函数callback(err, result)

例子：

var jlib = require('jingtum-lib');

var Remote = jlib.Remote;

var remote = new Remote({server: 'ws://ts5.jingtum.com:5020', local\_sign:true});

remote.connect(function(err, result) {  
 if (err) {  
 console.log('err:',err);  
 }else{  
 console.log(result);  
 }  
});

返回结果：

{

fee\_base: 10,

fee\_ref: 10,

hostid: 'CARE',

ledger\_hash: 'E39CD0668FF2647FD930B1E446E7861EF7B6D194737275A1101B6AF2C4D392EB',

ledger\_index: 9074471,

ledger\_time: 573902570,

load\_base: 256,

load\_factor: 256,

pubkey\_node: 'n9J1qmabAVgRfQ2DPXRzt58odeGsb5FiGCeg2iUCbZ5nRnWuMncc',

random: 'AD8112448B3B114880D8F65F3309654A4A6FE4590839D10A5C1E8AF602338F5D',

reserve\_base: 0,

reserve\_inc: 0,

server\_status: 'full',

validated\_ledgers: '266955-9074471'

}

返回结果说明：

|  |  |  |
| --- | --- | --- |
| 参数 | 类型 | 说明 |
| fee\_base | Integer | 基础费用(手续费计算公式因子) |
| fee\_ref | Integer | 引用费用(手续费计算公式因子) |
| hostid | String | 主机名 |
| ledger\_hash | String | 账本hash |
| ledger\_index | Integer | 区块高度 |
| ledger\_time | Integer | 账本关闭时间 |
| pubkey\_node | String | 节点公钥 |
| reserve\_base | Integer | 账号保留值 |
| reserve\_inc | Integer | 用户每次挂单或信任冻结数量 |
| server\_status | String | 服务器状态 |
| validated\_ledgers | String | 账本区间 |

## 关闭连接

每个Remote对象可以手动关闭连接。

方法：disconnect()

参数：无

例子：

var jlib = require('jingtum-lib');

var Remote = jlib.Remote;

var remote = new Remote({server: 'ws://ts5.jingtum.com:5020', local\_sign:true});

remote.connect(function(err, result) {  
 if (err) {  
 console.log('err:',err);  
 }else{  
 remote.disconnect(); //关闭连接  
 }  
});

## 请求底层服务器信息

首先通过本方法返回一个Request对象，然后通过submit方法获得井通底层的服务器信息，包含服务程序版本号version、该服务器缓存的账本区间ledgers、节点公钥node、服务器当前状态state。其中服务器当前状态包含可提供服务状态full和验证节点状态proposing。

方法：requestServerInfo()

参数：无

返回：Request对象

例子：

var jlib = require('jingtum-lib');

var Remote = jlib.Remote;

var remote = new Remote({server: 'ws://ts5.jingtum.com:5020', local\_sign:true});

remote.connect(function (err, result) {  
 if (err) {  
 console.log('err:', err);  
 } else {  
 var req = remote.requestServerInfo();  
 req.submit(function (err, result) {  
 if (err) {  
 console.log('err:', err);  
 }  
 else {  
 console.log('serverInfo:', result);  
 }  
 });  
 }  
});

返回结果：

{

version: '0.29.60',

ledgers: '4685487-8642747',

node: 'n94PahMpVKEBUGPn7ymoowELiXD1QWBuMUh7DpyCzJn5hzHMiqoj',

state: 'full 122:58:24'

}

返回结果说明：

|  |  |  |
| --- | --- | --- |
| 参数 | 类型 | 说明 |
| version | String | 服务器部署项目版本 |
| ledgers | String | 账本区间 |
| node | String | 节点公钥 |
| state | String | 服务器状态 |

## 获取最新账本信息

首先通过本方法返回一个Request对象，然后通过submit方法获得最新账本信息，包括区块高度(ledger\_index)与区块hash(ledger\_hash)。

方法：requestLedgerClosed()

参数：无

返回：Request对象

例子：

var jlib = require('jingtum-lib');

var Remote = jlib.Remote;

var remote = new Remote({server: 'ws://ts5.jingtum.com:5020', local\_sign:true});

remote.connect(function (err, result) {  
 if (err) {  
 console.log('err:', err);  
 } else {  
 var req = remote.requestLedgerClosed();  
 req.submit(function (err, result) {  
 if (err) {  
 console.log('err:', err);  
 }  
 else {  
 console.log('ledgerInfo:', result);  
 }  
 });  
 }  
});

返回结果：

{

ledger\_hash: '945DA5D61F9AA986C784B6353C7169BC981315CDBC05922C4F34E41B8A45B19F',

ledger\_index: 8642794

}

返回结果说明：

|  |  |  |
| --- | --- | --- |
| 参数 | 类型 | 说明 |
| ledger\_hash | String | 账本hash |
| ledger\_index | String | 账本高度/区块高度 |

## 获取某一账本具体信息

首先通过本方法返回一个Request对象，然后通过submit方法获得某一账本的具体信息。

方法：remote.requestLedger({ledger\_index:’8488670’,transactions:true});

参数：

|  |  |  |
| --- | --- | --- |
| 参数 | 类型 | 说明 |
| ledger\_index | String | 井通区块高度 |
| ledger\_hash | String | 井通区块hash(与上面ledger\_index二选其一) |
| transactions | Boolean | 是否返回账本上的交易记录hash，默认false |

注：整体参数是Object类型，当参数都不填时，默认返回最新账本信息。

返回：Request对象

例子：

var jlib = require('jingtum-lib');

var Remote = jlib.Remote;

var remote = new Remote({server: 'ws://ts5.jingtum.com:5020', local\_sign:true});

remote.connect(function(err, result) {  
 if (err) {  
 return console.log('err:',err);  
 }

//var req = remote.requestLedger({});//将默认返回最新账本信息  
 var req = remote.requestLedger({  
 ledger\_index: '8488670',   
 transactions: true   
 });  
 req.submit(function(err, result) {  
 if(err) {console.log('err:',err);}  
 else if(result){  
 console.log('res:', result);  
 }  
 });  
});

返回结果：

{

accepted: true,

account\_hash: 'E64648702393BD4AEEEAEBFA5144030874070C83D3079A5367FDF8282C45E1EB',

close\_time: 572790860,

close\_time\_human: '2018-Feb-24 12:34:20',

close\_time\_resolution: 10,

closed: true,

hash: 'BBFE0C3F25EE707F79A0E4B361A00B2F9254C7DC7AB6B8FED3A804B51F864392',

ledger\_hash: 'BBFE0C3F25EE707F79A0E4B361A00B2F9254C7DC7AB6B8FED3A804B51F864392',

ledger\_index: '8488670',

parent\_hash: 'AB78480BDA1B6BA3E328713FD477209D35D3665DC13F5AD38E3AA58CD9CA0DCA',

seqNum: '8488670',

totalCoins: '599999999996231320',

total\_coins: '599999999996231320',

transaction\_hash: 'B0E15F21E416E2C720E2F5C4FD2B0B9B6A3EA619ADA3461BDB1242D022273336',

transactions:

[ '4D09192722D460C29458E401EA7023855521EF3FF8564825474AC6E0EB9CB9A6' ]

}

返回结果说明：

|  |  |  |
| --- | --- | --- |
| 参数 | 类型 | 说明 |
| accepted | Boolean | 区块是否已经产生 |
| account\_hash | String | 状态hash树根 |
| close\_time | Integer | 关闭时间 |
| close\_time\_human | String | 关闭时间 |
| close\_time\_resolution | Integer | 关闭周期 |
| closed | Boolean | 账本是否已经关闭 |
| hash | String | 账本hash |
| ledger\_hash | String | 账本hash |
| ledger\_index | String | 账本高度/区块高度 |
| parent\_hash | String | 上一区块hash值 |
| seqNum | String | 账本高度/区块高度 |
| totalCoins | String | swt总量 |
| total\_coins | String | swt总量 |
| transaction\_hash | String | 交易hash树根 |
| transactions | Array | 该账本里的交易列表 |

## 查询某一交易具体信息

首先通过本方法返回一个Request对象，然后通过submit方法获得某一交易的具体信息。

方法：remote.requestTx({hash:’xxx’});

参数：

|  |  |  |
| --- | --- | --- |
| 参数 | 类型 | 说明 |
| hash | String | 交易hash |

返回：Request对象

例子：

var jlib = require('jingtum-lib');

var Remote = jlib.Remote;

var remote = new Remote({server: 'ws://ts5.jingtum.com:5020', local\_sign:true});

remote.connect(function(err, result) {  
 if (err) {  
 return console.log('err:',err);  
 }  
 var req = remote.requestTx({hash: '084C7823C318B8921A362E39C67A6FB15ADA5BCCD0C7E9A3B13485B1EF2A4313'});  
 req.submit(function(err, result) {  
 if(err) {console.log('err:',err);}  
 else if(result){  
 console.log('res:', result);  
 var fee = result.Fee/1000000;  
 console.log('关键信息：【 交易费:', fee, '】');  
 }  
 });  
});

返回结果：

{

Account: 'jDUjqoDZLhzx4DCf6pvSivjkjgtRESY62c',

Amount: '3000000',

Destination: 'jB7rxgh43ncbTX4WeMoeadiGMfmfqY2xLZ',

Fee: '10000',

Flags: 0,

Memos: [ { Memo: [Object] } ],

Sequence: 12938,

SigningPubKey: '021EAF8D79442EF3C27B6398DB39D37F9593F2E5953CF39E57E9A109D4875A

7017',

Timestamp: 572980620,

TransactionType: 'Payment',

TxnSignature: '30440220256C52A2C921D1D13FEC048BBFCB1F638470125244F848495E43C99

0590269FD02206B0D0D2CB465DB367DD3677C8BA3713DFD47BAA94D630573218DA707AA9310A5',

date: 572980630,

hash: '084C7823C318B8921A362E39C67A6FB15ADA5BCCD0C7E9A3B13485B1EF2A4313',

inLedger: 8507647,

ledger\_index: 8507647,

meta:

{ AffectedNodes: [ [Object], [Object], [Object] ],

TransactionIndex: 0,

TransactionResult: 'tesSUCCESS' },

validated: true

}

返回结果说明：

|  |  |  |  |
| --- | --- | --- | --- |
| 参数 | | 类型 | 说明 |
| Account | | String | 钱包地址 |
| AppType | | Integer | 应用来源（正整数，挂单中设置了app时，此参数才有） |
| Amount | | String/Object | 交易金额 |
| Destination | | String | 交易对家地址 |
| Fee | | String | 交易费 |
| Flags | | Integer | 交易标记 |
| Memos | | Array | 备注 |
| Sequence | | Integer | 自身账号的交易号 |
| SigningPubKey | | String | 签名公钥 |
| Timestamp | | Integer | 交易提交时间戳 |
| TransactionType | | String | 交易类型 |
| TxnSignature | | String | 交易签名 |
| date | | Integer | 交易进账本时间 |
| hash | | String | 交易hash |
| inLedger | | Integer | 交易所在的账本号 |
| ledger\_index | | Integer | 账本高度 |
| meta | | Object | 交易影响的节点 |
|  | AffectedNodes | Array | 受影响的节点 |
| TransactionIndex | Integer | -- |
| TransactionResult | String | 交易结果 |
| validated | | Boolean | 交易是否通过验证 |

## 请求账号信息

首先通过本方法返回一个Request对象，然后通过submit方法获得某一账号的交易信息。

方法：remote.requestAccountInfo({account:’xxx’});

参数：

|  |  |  |
| --- | --- | --- |
| 参数 | 类型 | 说明 |
| account | String | 井通钱包地址 |

返回：Request对象

例子：

var jlib = require('jingtum-lib');

var Remote = jlib.Remote;

var remote = new Remote({server: 'ws://ts5.jingtum.com:5020', local\_sign:true});

remote.connect(function(err, result) {  
 if (err) {  
 return console.log('err:',err);  
 }  
 var options = {account: 'jB7rxgh43ncbTX4WeMoeadiGMfmfqY2xLZ'};  
 var req = remote.requestAccountInfo(options);  
 req.submit(function(err, result) {  
 if(err) {console.log('err:',err);}  
 else if(result){  
 console.log('res:', result);  
 }  
 });  
});

返回结果：

{

account\_data:

{ Account: 'jB7rxgh43ncbTX4WeMoeadiGMfmfqY2xLZ',

Balance: '10884068241',

Domain: '687474703A2F2F7777772E6A696E6774756D2E636F6D2F',

Flags: 0,

LedgerEntryType: 'AccountRoot',

MessageKey: '736574206D657373616765206B65792074657374',

OwnerCount: 43,

PreviousTxnID: 'AE11268454C05B4A2FFFBEC53FFC8FC92118DE387B2911A00879E6BD76C27D96',

PreviousTxnLgrSeq: 8642907,

RegularKey: 'jp53tPyrQLoFriTJhtm8Z9iLUXUDucnwVk',

Sequence: 1934,

TransferRate: 1800000000,

index: 'E80FF91725E82A623ADC46B458D37FB270651A76A07CD6C8115F1202856342E6' },

ledger\_hash: '79F1CB1EEF303E8928635AA68E4E0319898B7C113AC4C183CD591F274F0FE2D4',

ledger\_index: 8643224,

validated: true

}

返回结果说明：

|  |  |  |  |
| --- | --- | --- | --- |
| 参数 | | 类型 | 说明 |
| account\_data | | Object | 账号信息 |
|  | Account | String | 钱包地址 |
| Balance | String | swt数量 |
| Domain | String | 域名 |
| Flags | Integer | 属性标志 |
| MessageKey | String | 公共密钥，用于发送加密的邮件到这个帐户 |
| OwnerCount | Integer | 用户拥有的挂单数和信任线数量的总和 |
| PreviousTxnID | String | 操作该帐号的上一笔交易hash |
| PreviousTxnLgrSeq | Integer | 该帐号上一笔交易所在的账本号 |
| RegularKey | String | RegularKey |
| Sequence | Integer | 账号当前序列号 |
| TransferRate | Integer | 手续费汇率 |
| index | String | 该数据所在索引hash |
| ledger\_hash | | String | 账本hash |
| ledger\_index | | Integer | 账本高度 |
| validated | | Boolean | 交易是否通过验证 |

## 获得账号可接收和发送的货币

首先通过本方法返回一个Request对象，然后通过submit方法获得某一账号可发送和接收的货币种类。

方法：remote.requestAccountTums({account:’xxx’});

参数：

|  |  |  |
| --- | --- | --- |
| 参数 | 类型 | 说明 |
| account | String | 井通钱包地址 |

返回：Request对象

例子：

var jlib = require('jingtum-lib');

var Remote = jlib.Remote;

var remote = new Remote({server: 'ws://ts5.jingtum.com:5020', local\_sign:true});

remote.connect(function(err, result) {  
 if (err) {  
 return console.log('err:',err);  
 }  
 var options = {account: 'jB7rxgh43ncbTX4WeMoeadiGMfmfqY2xLZ'};  
 var req = remote.requestAccountTums(options);  
 req.submit(function(err, result) {  
 if(err) {console.log('err:',err);}  
 else if(result){  
 console.log('res:', result);  
 }  
 });  
});

返回结果：

{

ledger\_hash: '870FD292E341D25CEEC919E68FAE4C0E40F6E46045226EED3793B34F08030688',

ledger\_index: 8643336,

receive\_currencies:

[ 'AAA',

'CCA',

'CNY',

'USD',

'8100000001000020160022000000000020000001',

'8100000036000020160622201606300120000002',

'8300000001000020160021201600240020000001',

'830000000B000020160004000000000020000001' ],

send\_currencies:

[ 'AAA',

'CCA',

'CNY',

'USD',

'8100000001000020160022000000000020000001',

'8100000036000020160622201606300120000002',

'8300000001000020160021201600240020000001',

'830000000B000020160004000000000020000001' ],

validated: true

}

返回结果说明：

|  |  |  |
| --- | --- | --- |
| 参数 | 类型 | 说明 |
| ledger\_hash | String | 账本hash |
| ledger\_index | Integer | 账本高度 |
| receive\_currencies | Array | 可接收的货币列表 |
| send\_currencies | Array | 可发送的货币列表 |
| validated | Boolean | 交易是否通过验证 |

## 获得账号关系

井通账户之间会建立各种不同的关系。这些关系由井通后台的关系（relations）机制来处理，目前支持以下关系：信任(trust)、授权(authorize)、冻结(freeze)。

首先通过本方法返回一个Request对象，然后通过submit方法获得某一账号指定关系的信息。

方法：remote.requestAccountRelations({account:’xxx’,type:’xxx’});

参数：

|  |  |  |
| --- | --- | --- |
| 参数 | 类型 | 说明 |
| account | String | 井通钱包地址 |
| type | String | 关系类型，固定的三个值：trust、authorize、freeze |

返回：Request对象

例子：

var jlib = require('jingtum-lib');

var Remote = jlib.Remote;

var remote = new Remote({server: 'ws://ts5.jingtum.com:5020', local\_sign:true});

remote.connect(function(err, result) {  
 if (err) {  
 return console.log('err:',err);  
 }  
 var options = {account: 'jB7rxgh43ncbTX4WeMoeadiGMfmfqY2xLZ',type:'trust'};  
 var req = remote.requestAccountRelations(options);  
 req.submit(function(err, result) {  
 if(err) {console.log('err:',err);}  
 else if(result){  
 console.log('res:', result);  
 }  
 });  
});

返回结果：

{

account: 'jB7rxgh43ncbTX4WeMoeadiGMfmfqY2xLZ',

ledger\_hash: 'EFC95414F74F4DFC27703E68FAB28186818A5F596D5CD105EC295B34C7E5A70D',

ledger\_index: 8643385,

lines:

[ { account: 'jBciDE8Q3uJjf111VeiUNM775AMKHEbBLS',

balance: '7.00193333333334',

currency: 'USD',

limit: '100',

limit\_peer: '0',

quality\_in: 0,

quality\_out: 0 },

{ account: 'jGRqbsiAf2vgdbet5z31hNLEt8hdBjfPUk',

balance: '-10',

currency: 'AAA',

limit: '100',

limit\_peer: '1000',

no\_skywell\_peer: true,

quality\_in: 0,

quality\_out: 0 },

{ account: 'jBciDE8Q3uJjf111VeiUNM775AMKHEbBLS',

balance: '133454.4061014615',

currency: 'CNY',

limit: '10000000000',

limit\_peer: '0',

no\_skywell: true,

quality\_in: 0,

quality\_out: 0 },

{ account: 'jBciDE8Q3uJjf111VeiUNM775AMKHEbBLS',

balance: '999984',

currency: '8100000036000020160622201606300120000002',

limit: '10000000000',

limit\_peer: '0',

no\_skywell: true,

quality\_in: 0,

quality\_out: 0 },

{ account: 'jMcCACcfG37xHy7FgqHerzovjLM5FCk7tT',

balance: '0.04995',

currency: 'USD',

limit: '10000000000',

limit\_peer: '0',

no\_skywell: true,

quality\_in: 0,

quality\_out: 0 } ],

validated: true

}

返回结果说明：

|  |  |  |  |
| --- | --- | --- | --- |
| 参数 | | 类型 | 说明 |
| account | | String | 钱包地址 |
| ledger\_hash | | String | 账本hash |
| ledger\_index | | Integer | 账本高度 |
| lines | | Array | 该账户的信任线 |
|  | account | String | 信任的银关 |
| balance | String | 余额 |
| currency | String | 货币种类 |
| limit | String | 信任额度 |
| limit\_peer | String | 对方设置的信任额度，默认0 |
| quality\_in | Integer | 兑换比例，默认0，暂时未用 |
| quality\_out | Integer | 兑换比例，默认0，暂时未用 |
| validated | | Boolean | 交易是否通过验证 |

## 获得账号挂单

首先通过本方法返回一个Request对象，然后通过submit方法获得某一账号的挂单信息。

方法：remote.requestAccountOffers({account:’xxx’});

参数：

|  |  |  |
| --- | --- | --- |
| 参数 | 类型 | 说明 |
| account | String | 井通钱包地址 |

返回：Request对象

例子：

var jlib = require('jingtum-lib');

var Remote = jlib.Remote;

var remote = new Remote({server: 'ws://ts5.jingtum.com:5020', local\_sign:true});

remote.connect(function(err, result) {  
 if (err) {  
 return console.log('err:',err);  
 }  
 var options = {account: 'jB7rxgh43ncbTX4WeMoeadiGMfmfqY2xLZ'};  
 var req = remote.requestAccountOffers(options);  
 req.submit(function(err, result) {  
 if(err) {console.log('err:',err);}  
 else if(result){  
 console.log('res:', result);  
 }  
 });  
});

返回结果：

{

account: 'jB7rxgh43ncbTX4WeMoeadiGMfmfqY2xLZ',

ledger\_hash: 'A88647AE9459F2E9538190D857D82437003C21769DE22D6A2F041A33F83E2753',

ledger\_index: 8643476,

offers:

[ { AppType: 3,

FeeCurrency: 'VCC',

OfferFeeRateDen: '1000',

OfferFeeRateNum: '3',

flags: 131072,

seq: 206,

taker\_gets: {

currency: 'USD',

issuer: 'jBciDE8Q3uJjf111VeiUNM775AMKHEbBLS',

value: '0.5' },

taker\_pays: '10000000' },

{ AppType: 3,

FeeCurrency: 'VCC',

OfferFeeRateDen: '1000',

OfferFeeRateNum: '3',

flags: 131072,

seq: 1859,

taker\_gets: {

currency: 'USD',

issuer: 'jBciDE8Q3uJjf111VeiUNM775AMKHEbBLS',

value: '0.5' },

taker\_pays: '1000000' } ],

validated: true

}

返回结果说明：

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 参数 | | | 类型 | 说明 |
| account | | | String | 钱包地址 |
| ledger\_hash | | | String | 账本hash |
| ledger\_index | | | Integer | 账本高度 |
| offers | | | Array | 该账户的挂单列表 |
|  | AppType | | String | 应用来源序号 |
| FeeCurrency | | String | 货币种类 |
| OfferFeeRateDen | | String | 分母 |
| OfferFeeRateNum | | String | 分子 |
| flags | | Integer | 买卖类型（131072表示卖，否则是买） |
| seq | | String | 余额 |
| taker\_gets | | String | 货币种类 |
|  | value | String | 金额 |
| currency | String | 货币种类 |
| issuer | String | 货币 |
| taket\_pays | | String | 信任额度 |
|  | value | String | 金额 |
| currency | String | 货币种类 |
| issuer | String | 货币 |
| validated | | | Boolean | 交易是否通过验证 |

## 获得账号交易列表

首先通过本方法返回一个Request对象，然后通过submit方法获得某一账号的交易列表信息。

方法：remote.requestAccountTx({account:’xxx’});

参数：

|  |  |  |
| --- | --- | --- |
| 参数 | 类型 | 说明 |
| account | String | 井通钱包地址 |
| limit | Integer | 限定返回多少条记录，默认200 |

返回：Request对象

例子：

var jlib = require('jingtum-lib');

var Remote = jlib.Remote;

var remote = new Remote({server: 'ws://ts5.jingtum.com:5020', local\_sign:true});

remote.connect(function(err, result) {  
 if (err) {  
 return console.log('err:',err);  
 }  
 var options = {account: 'jB7rxgh43ncbTX4WeMoeadiGMfmfqY2xLZ'};  
 var req = remote.requestAccountTx(options);  
 req.submit(function(err, result) {  
 if(err) {console.log('err:',err);}  
 else if(result){  
 console.log('res:', result);  
 }  
 });  
});

返回结果：

{

account: 'jB7rxgh43ncbTX4WeMoeadiGMfmfqY2xLZ',

ledger\_index\_max: 8643603,

ledger\_index\_min: 4685487,

marker: { ledger: 8478562, seq: 60 },

transactions:

[ { date: 1521021610,

hash: 'EDC7497E0E7FF3ED79F6A3829401B71F4446C3521E17AC40A937DC3DC9023CF6',

type: 'received',

fee: '0.01',

result: 'tesSUCCESS',

memos: [Array],

counterparty: 'jDUjqoDZLhzx4DCf6pvSivjkjgtRESY62c',

amount: [Object],

effects: [],

balances: [Object]},

... 100 more items ]

}

返回结果说明：

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 参数 | | | 类型 | 说明 |
| account | | | String | 钱包地址 |
| ledger\_index\_max | | | Integer | 当前节点缓存的账本区间最大值 |
| ledger\_index\_min | | | Integer | 当前节点缓存的账本区间最小值 |
| marker | | | Object | 查到的当前记录标记 |
| transactions | | | Array | 交易记录列表 |
|  | date | | Integer | 时间戳 |
| hash | | String | 交易hash |
| type | | String | 交易类型 |
| fee | | String | 手续费 |
| result | | String | 交易结果 |
| memos | | Array | 备注 |
| counterparty | | String | 交易对家 |
| amount | | Object | 交易金额对象 |
|  | value | String | 金额 |
| currency | String | 货币种类 |
| issuer | String | 货币 |
| effects | | Array | 交易效果，详见后面effects说明。 |
| balances | | Object | 每笔交易后变化的币种余额 |

## 获得市场挂单列表

首先通过本方法返回一个Request对象，然后通过submit方法获得市场挂单列表信息。

方法：remote.requestOrderBook({});

参数：

|  |  |  |
| --- | --- | --- |
| 参数 | 类型 | 说明 |
| gets | Object | 对家想要获得的货币信息 |
| pays | Object | 对家想要支付的货币信息 |

返回：Request对象

例子：

var jlib = require('jingtum-lib');

var Remote = jlib.Remote;

var remote = new Remote({server: 'ws://ts5.jingtum.com:5020', local\_sign:true});

remote.connect(function(err, result) {  
 if (err) {  
 return console.log('err:',err);  
 }  
 var options = {

gets: { currency: 'SWT', issuer: '' },

pays: { currency: 'CNY', issuer: 'jBciDE8Q3uJjf111VeiUNM775AMKHEbBLS' }};

var req = remote.requestOrderBook(options);  
 req.submit(function(err, result) {  
 if(err) {console.log('err:',err);}  
 else if(result){  
 console.log('res:', result);  
 }  
 });  
});

返回结果：

{

ledger\_current\_index: 8649275,

offers:

[ { Account: 'j9eM8GiBb4QFRZZsrsde6XTPDenXEFnrkm',

BookDirectory: '61EF253552DDD8E9D5A1071C81591086E0132391F92311BB5B03BD5CE35839AA',

BookNode: '0000000000000000',

Flags: 0,

LedgerEntryType: 'Offer',

OwnerNode: '0000000000000000',

PreviousTxnID: '547266B62E7B0931F829C58B89541BCFE94ECB1EA78E712D8DAD071D40A1EDB9',

PreviousTxnLgrSeq: 8574138,

Sequence: 11,

TakerGets: [Object],

TakerPays: '1028266745029520',

index: 'CDD0F0DE4D63DB7F208B6EF5A84D2FE5A8CDD53DB68B876A17E9EE357EFC877B',

owner\_funds: '988270574.4452997',

quality: '1052631.578982826' },

{ Account: 'jKog7BTbU7wzDDDH3FPPmveszWQnZSeP5W',

BookDirectory: '61EF253552DDD8E9D5A1071C81591086E0132391F92311BB5B03C78C2C91CEFB',

BookNode: '0000000000000000',

Flags: 0,

LedgerEntryType: 'Offer',

OwnerNode: '0000000000000000',

PreviousTxnID: 'BBB6A770A818427EC3B3778F94EF1577704EFCAD0C7602740E304207E9FC4943',

PreviousTxnLgrSeq: 1352529,

Sequence: 10,

TakerGets: [Object],

TakerPays: '100000000',

index: '9EDE0EDF25FA1AAC26B11DB38F90095AC8A84351B4B83773162A5B76BFF98674',

owner\_funds: '1001841.261900998',

quality: '1063829.787234043' },

... 200 more items ],

validated: false

}

返回结果说明：

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 参数 | | | 类型 | 说明 |
| ledger\_current\_index | | | String | 当前账本号 |
| offers | | | Array | 市场挂单列表 |
|  | Account | | Integer | 账号地址 |
| BookDirectory | | String | -- |
| BookNode | | String | -- |
| Flags | | Integer | 挂单买卖标记 |
| LedgerEntryType | | String | 账本数据结构类型 |
| OwnerNode | | Array | -- |
| PreviousTxnID | | String | 上一笔交易hash |
| PreviousTxnLgrSeq | | Integer | 上一笔交易所在账本号 |
| Sequence | | Integer | 单子序列号 |
| TakerGets | | Object | 对方得到的。（买卖双方，当货币是swt时，数据类型为对象；否则为string） |
|  | value | String | 金额 |
| currency | String | 货币种类 |
| issuer | String | 货币 |
| TakerPays | | String | 对方支付的 |
| index | | String | 该数据所在索引hash |
| owner\_funds | | String | 用户swt资产 |
| quality | | String | 价格或价格的倒数 |
| validated | | | Boolean | 交易是否通过验证 |

## 获得挂单佣金设置信息

首先通过本方法返回一个Request对象，然后通过submit方法获得市场挂单列表信息。

方法：remote.requestBrokerage({});

参数：

|  |  |  |
| --- | --- | --- |
| 参数 | 类型 | 说明 |
| issuer | String | 货币发行方 |
| app | Integer | 应用来源 |
| currency | String | 货币种类 |

返回：Request对象

例子：

var jlib = require('jingtum-lib');

var Remote = jlib.Remote;

var remote = new Remote({server: 'ws://ts5.jingtum.com:5020', local\_sign:true});

remote.connect(function(err, result) {  
 if (err) {  
 return console.log('err:',err);  
 }  
 var options = {

issuer: 'jBciDE8Q3uJjf111VeiUNM775AMKHEbBLS',

app: 1

currency: 'TES'};

var req = remote.requestBrokerage(options);  
 req.submit(function(err, result) {  
 if(err) {console.log('err:',err);}  
 else if(result){  
 console.log('res:', result);  
 }  
 });  
});

返回结果：

{

AppType: '1',

FeeCurrency: 'TES',

FeeCurrencyIssuer: 'jBciDE8Q3uJjf111VeiUNM775AMKHEbBLS',

OfferFeeRateDen: '1000',

OfferFeeRateNum: '3',

ledger\_hash: 'EEEECA6766A17D4113CDF2FC6DA05C2B735EDFB437EF2546DEDBA127AF63B2BE',

ledger\_index: 1633,

rate\_den: '1000',

rate\_num: '1',

validated: true

}

返回结果说明：

|  |  |  |
| --- | --- | --- |
| 参数 | 类型 | 说明 |
| AppType | String | 应用来源序号 |
| FeeCurrency | String | 货币种类 |
| FeeCurrencyIssuer | String | 货币发行方 |
| OfferFeeRateDen | String | 分母 |
| OfferFeeRateNum | String | 分子 |
| ledger\_hash | String | 当前hash |
| ledger\_index | Integer | 当前账本号 |
| rate\_den | String | 分母 |
| rate\_num | String | 分子 |
| validated | Boolean | 交易是否通过验证 |

## 支付

首先通过buildPaymentTx方法返回一个Transaction对象，然后通过setSecret传入密钥，addMemo添加备注为可选项，最后通过submit方法提交支付信息。

### 创建支付对象

方法：remote.buildPaymentTx({});

参数：

|  |  |  |  |
| --- | --- | --- | --- |
| 参数 | | 类型 | 说明 |
| account | | String | 发起账号 |
| to | | String | 目标账号 |
| amount | | Object | 支付金额 |
|  | value | String | 支付数量 |
| currency | String | 货币种类，三到六个字母或20字节的自定义货币 |
| issuer | String | 货币发行方 |

返回：Transaction对象

### 传入密钥

方法：tx.setSecret(secret);

参数：

|  |  |  |
| --- | --- | --- |
| 参数 | 类型 | 说明 |
| secret | String | 井通钱包私钥 |

### 设置备注

方法：tx.addMemo(memo);

参数：

|  |  |  |
| --- | --- | --- |
| 参数 | 类型 | 说明 |
| memo | String | 备注信息 |

### 提交支付

方法：tx.submit(callback);

参数：无

支付完整例子：

var jlib = require('jingtum-lib');

var Remote = jlib.Remote;

var remote = new Remote({server: 'ws://ts5.jingtum.com:5020', local\_sign:true});

remote.connect(function(err, result) {  
 if (err) {  
 return console.log('err:',err);  
 }  
 var tx = remote.buildPaymentTx({  
 account: 'jB7rxgh43ncbTX4WeMoeadiGMfmfqY2xLZ',  
 to: 'jDUjqoDZLhzx4DCf6pvSivjkjgtRESY62c',  
 amount: {  
 "value": 0.5,  
 "currency": "SWT",  
 "issuer": ""  
 }  
 });  
 tx.setSecret('sn37nYrQ6KPJvTFmaBYokS3FjXUWd');  
 tx.addMemo('给jDUjqoDZLhzx4DCf6pvSivjkjgtRESY62c支付0.5swt.');//可选  
 tx.submit(function(err, result) {  
 if(err) {console.log('err:',err);}  
 else if(result){  
 console.log('res:', result);  
 }  
 });  
});

返回结果：

{

engine\_result: 'tesSUCCESS',

engine\_result\_code: 0,

engine\_result\_message: 'The transaction was applied. Only final in a validated ledger.',

tx\_blob: '1200002200000000240000079261400000000007A120684000000000002710732102FE64E0C20F0058F22F3742EDC15F49F318C04F88B130742C68BAF3B1C89FD1677446304402202EC74EFA5D3AAA663B49610B45E92438C5FD206376CB23323726180656C9EFF9022038577D38CA50296A198A4CD2BE06BCFD8916A64E40D68F5C91C12110A9246698811472F05993EBA9858291D364EBF6EEC3D851BD3792831485B6C98BAD6DBF7805D3C5CCC1B4F989E0CE6749F9EA7D32E7BB996A44556A716F445A4C687A78344443663670765369766A6B6A677452455359363263E694AFE4BB98302E357377742EE1F1',

tx\_json:

{ Account: 'jB7rxgh43ncbTX4WeMoeadiGMfmfqY2xLZ',

Amount: '500000',

Destination: 'jDUjqoDZLhzx4DCf6pvSivjkjgtRESY62c',

Fee: '10000',

Flags: 0,

Memos: [ [Object] ],

Sequence: 1938,

SigningPubKey: '02FE64E0C20F0058F22F3742EDC15F49F318C04F88B130742C68BAF3B1C89FD167',

TransactionType: 'Payment',

TxnSignature: '304402202EC74EFA5D3AAA663B49610B45E92438C5FD206376CB23323726180656C9EFF9022038577D38CA50296A198A4CD2BE06BCFD8916A64E40D68F5C91C12110A9246698',

hash: 'FA886D470C6B3A96FBBD57639325AEFB9104B56D912AE8AE9E385914EA63E6B6' }

}

返回结果说明：

|  |  |  |  |
| --- | --- | --- | --- |
| 参数 | | 类型 | 说明 |
| engine\_result | | String | 请求结果 |
| engine\_result\_code | | Array | 请求结果编码 |
| engine\_result\_message | | String | 请求结果message信息 |
| tx\_blob | | String | 16进制签名后的交易 |
| tx\_json | | Object | 交易内容 |
|  | Account | String | 账号地址 |
| Amount | String | 交易金额 |
| Destination | String | 对家 |
| Fee | String | 交易费 |
| Flags | Integer | 交易标记 |
| Memos | Array | 备注 |
| Sequence | Integer | 单子序列号 |
| SigningPubKey | Object | 签名公钥 |
| TransactionType | String | 交易类型 |
| TxnSignature | String | 交易签名 |
| hash | String | 交易hash |

## 设置关系

首先通过buildRelationTx方法返回一个Transaction对象，然后通过setSecret传入密钥，最后通过submit方法提交支付信息。目前支持的关系类型：信任(trust)、授权(authorize)、冻结(freeze)。

### 创建关系对象

方法：remote.buildRelationTx({});

参数：

|  |  |  |  |
| --- | --- | --- | --- |
| 参数 | | 类型 | 说明 |
| type | | String | 关系种类 |
| account | | String | 设置关系的源账号 |
| target | | String | 目标账号，授权和冻结才有 |
| limit | | Object | 关系金额 |
|  | value | String | 数量 |
| currency | String | 货币种类，三到六个字母或20字节的自定义货币 |
| issuer | String | 货币发行方 |

返回：Transaction对象

### 传入密钥

方法：tx.setSecret(secret);

参数：

|  |  |  |
| --- | --- | --- |
| 参数 | 类型 | 说明 |
| secret | String | 井通钱包私钥 |

### 关系设置

方法：tx.submit(callback);

参数：无

设置关系完整例子：

var jlib = require('jingtum-lib');

var Remote = jlib.Remote;

var remote = new Remote({server: 'ws://ts5.jingtum.com:5020'});

remote.connect(function(err, result) {

if (err) {

return console.log('err:',err);

}

var options = {

account: 'jB7rxgh43ncbTX4WeMoeadiGMfmfqY2xLZ',

target: 'jDUjqoDZLhzx4DCf6pvSivjkjgtRESY62c',

limit:{

currency: 'CCA',

value: "0.01",

issuer: 'js7M6x28mYDiZVJJtfJ84ydrv2PthY9W9u'

},

type:'authorize'

};

var tx = remote.buildRelationTx(options);

tx.setSecret('sn37nYrQ6KPJvTFmaBYokS3FjXUWd');

tx.submit(function(err, result) {

if(err) {console.log('err:',err);}

else if(result){

console.log('res:', result);

}

});

});

返回结果：

{

engine\_result: 'tesSUCCESS',

engine\_result\_code: 0,

engine\_result\_message: 'The transaction was applied. Only final in a validated ledger.',

tx\_blob: '120015220000000024000007932F223CA4FC20230000000163D4038D7EA4C6800000000000000000000000000043434100000000001B1CC46616D04D5A8EB8397A7C60B623521336DA684000000000002710732102FE64E0C20F0058F22F3742EDC15F49F318C04F88B130742C68BAF3B1C89FD16774463044022064FE815BAB29F691BC5515AB409AF5866A75FA474C0018F33B0D2CF1500170BE02202EA9CA5DAD91922A9785C8837C9C1037E0BEB1E0818EA63FA102F558C6ED126C811472F05993EBA9858291D364EBF6EEC3D851BD3792871485B6C98BAD6DBF7805D3C5CCC1B4F989E0CE6749',

tx\_json:

{ Account: 'jB7rxgh43ncbTX4WeMoeadiGMfmfqY2xLZ',

Fee: '10000',

Flags: 0,

LimitAmount:

{ currency: 'CCA',

issuer: 'js7M6x28mYDiZVJJtfJ84ydrv2PthY9W9u',

value: '0.01' },

RelationType: 1,

Sequence: 1939,

SigningPubKey: '02FE64E0C20F0058F22F3742EDC15F49F318C04F88B130742C68BAF3B1C89FD167',

Target: 'jDUjqoDZLhzx4DCf6pvSivjkjgtRESY62c',

Timestamp: 574399740,

TransactionType: 'RelationSet',

TxnSignature: '3044022064FE815BAB29F691BC5515AB409AF5866A75FA474C0018F33B0D2CF1500170BE02202EA9CA5DAD91922A9785C8837C9C1037E0BEB1E0818EA63FA102F558C6ED126C',

hash: '2BC2051D97999852E515A0E5733F3C0FCC21E0270D93C5CEB43D80BDC175735A' }

}

返回结果说明：

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 参数 | | | 类型 | 说明 |
| engine\_result | | | String | 请求结果 |
| engine\_result\_code | | | Array | 请求结果编码 |
| engine\_result\_message | | | String | 请求结果message信息 |
| tx\_blob | | | String | 16进制签名后的交易 |
| tx\_json | | | Object | 交易内容 |
|  | Account | | String | 账号地址 |
| Fee | | String | 交易费 |
| Flags | | Integer | 交易标记 |
| LimitAmount | | Object | 关系的额度 |
|  | currency | String | 货币 |
| issuer | String | 货币发行方 |
| value | String | 额度 |
| RelationType | | Integer | 关系类型：0信任；1授权；3冻结/解冻； |
| Sequence | | Integer | 单子序列号 |
| SigningPubKey | | Object | 签名公钥 |
| Target | | String | 关系对家 |
| Timestamp | | Integer | 时间戳 |
| TransactionType | | String | 交易类型：TrustSet信任;RelationDel解冻；RelationSet授权/冻结 |
| TxnSignature | | String | 交易签名 |
| hash | | String | 交易hash |

## 设置账号属性 -----------待完善

首先通过buildAccountSetTx方法返回一个Transaction对象，然后通过setSecret传入密钥，最后通过submit方法设置账号属性。目前支持的三类：`property`、 `delegate` 、`signer`。property用于设置账号一般属性；delegate用于某账号设置委托帐户；signer用于设置签名。

### 创建属性对象

方法：remote.buildAccountSetTx({});

参数：

|  |  |  |
| --- | --- | --- |
| 参数 | 类型 | 说明 |
| type | String | 属性种类 |
| account | String | 设置属性的源账号 |
| set\_flag | String | 属性编号 |

返回：Transaction对象

### 传入密钥

方法：tx.setSecret(secret);

参数：

|  |  |  |
| --- | --- | --- |
| 参数 | 类型 | 说明 |
| secret | String | 井通钱包私钥 |

### 属性设置

方法：tx.submit(callback);

参数：无

设置属性完整例子：

var jlib = require('jingtum-lib');

var Remote = jlib.Remote;

var remote = new Remote({server: 'ws://ts5.jingtum.com:5020'});

remote.connect(function(err, result) {

if (err) {

return console.log('err:',err);

}

var options = {

account: 'jB7rxgh43ncbTX4WeMoeadiGMfmfqY2xLZ',

type:'property'

};

var tx = remote.buildAccountSetTx(options);

tx.setSecret('sn37nYrQ6KPJvTFmaBYokS3FjXUWd');

tx.submit(function(err, result) {

if(err) {console.log('err:',err);}

else if(result){

console.log('res:', result);

}

});

});

## 挂单

首先通过buildOfferCreateTx方法返回一个Transaction对象，然后通过setSecret传入密钥，最后通过submit方法提交挂单。

### 创建挂单对象

方法：remote.buildOfferCreateTx({});

参数：

|  |  |  |  |
| --- | --- | --- | --- |
| 参数 | | 类型 | 说明 |
| type | | String | 挂单类型，固定的两个值：Buy、Sell |
| account | | String | 挂单方账号 |
| app | | Integer | 应用来源序号（正整数），可选 |
| taker\_gets | | Object | 对方得到的，即挂单方支付的 |
|  | value | String | 数量 |
| currency | String | 货币种类 |
| issuer | String | 货币发行方 |
| taker\_pays | | Object | 对方支付的，即挂单方获得的 |
|  | value | String | 数量 |
| currency | String | 货币种类 |
| issuer | String | 货币发行方 |

返回：Transaction对象

### 传入密钥

方法：tx.setSecret(secret);

参数：

|  |  |  |
| --- | --- | --- |
| 参数 | 类型 | 说明 |
| secret | String | 井通钱包私钥 |

### 提交挂单

方法：tx.submit(callback);

参数：无

挂单完整例子：

var jlib = require('jingtum-lib');

var Remote = jlib.Remote;

var remote = new Remote({server: 'ws://ts5.jingtum.com:5020', local\_sign:true});

remote.connect(function (err, result) {

if (err) {

return console.log('err:', err);

}

var options = {

type: 'Sell',

account: 'jB7rxgh43ncbTX4WeMoeadiGMfmfqY2xLZ',

taker\_gets: {

value: '0.01',

currency: 'CNY',

issuer: 'jBciDE8Q3uJjf111VeiUNM775AMKHEbBLS'

},

taker\_pays: {

value: '1',

currency: 'SWT',

issuer: ''

}

};

var tx = remote.buildOfferCreateTx(options);

tx.setSecret('sn37nYrQ6KPJvTFmaBYokS3FjXUWd');

tx.submit(function (err, result) {

if (err) {

console.log('err:', err);

}

else if (result) {

console.log('res:', result);

}

});

});

返回结果：

{

engine\_result: 'tesSUCCESS',

engine\_result\_code: 0,

engine\_result\_message: 'The transaction was applied. Only final in a validated ledger.',

tx\_blob: '120007220008000024000007982F223CAB026440000000000F424065D4038D7EA4C68000000000000000000000000000434E5900000000007478E561645059399B334448F7544F2EF308ED32684000000000002710732102FE64E0C20F0058F22F3742EDC15F49F318C04F88B130742C68BAF3B1C89FD1677446304402205E9C26D6F0C4ECC043FAC2D7ADA9740763C89B999857C56047BD3AD3F758976A0220501A401B89560B1F4FA2E9E7727674C0882AA4935748B3A4CBC5BA63F17CCE40811472F05993EBA9858291D364EBF6EEC3D851BD3792',

tx\_json:

{ Account: 'jB7rxgh43ncbTX4WeMoeadiGMfmfqY2xLZ',

Fee: '10000',

Flags: 524288,

Sequence: 1944,

SigningPubKey: '02FE64E0C20F0058F22F3742EDC15F49F318C04F88B130742C68BAF3B1C89FD167',

TakerGets:

{ currency: 'CNY',

issuer: 'jBciDE8Q3uJjf111VeiUNM775AMKHEbBLS',

value: '0.01' },

TakerPays: '1000000',

Timestamp: 574401282,

TransactionType: 'OfferCreate',

TxnSignature: '304402205E9C26D6F0C4ECC043FAC2D7ADA9740763C89B999857C56047BD3AD3F758976A0220501A401B89560B1F4FA2E9E7727674C0882AA4935748B3A4CBC5BA63F17CCE40',

hash: 'F6B113B1513CE6F5B7CB130028FD06E564E35CC7E83876460E9A9F0038365424' }

}

返回结果说明：

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 参数 | | | 类型 | 说明 |
| engine\_result | | | String | 请求结果 |
| engine\_result\_code | | | Array | 请求结果编码 |
| engine\_result\_message | | | String | 请求结果message信息 |
| tx\_blob | | | String | 16进制签名后的交易 |
| tx\_json | | | Object | 交易内容 |
|  | Account | | String | 账号地址 |
| Fee | | String | 交易费 |
| AppType | | Integer | 应用来源标记（正整数，当挂单中设置了app时，此参数才有） |
| Flags | | Integer | 交易标记 |
| Sequence | | Integer | 单子序列号 |
| SigningPubKey | | String | 签名公钥 |
| TakerGets | | Object | 对家得到的 |
|  | currency | String | 货币 |
| issuer | String | 货币发行方 |
| value | String | 额度 |
| TakerPays | | String | 对家支付的； |
| Timestamp | | Integer | 时间戳 |
| TransactionType | | String | 交易类型：TrustSet信任;RelationDel解冻；RelationSet授权/冻结 |
| TxnSignature | | String | 交易签名 |
| hash | | String | 交易hash |

## 取消挂单

首先通过buildOfferCancelTx方法返回一个Transaction对象，然后通过setSecret传入密钥，最后通过submit方法取消挂单。

### 创建取消挂单对象

方法：remote.buildOfferCancelTx({});

参数：

|  |  |  |
| --- | --- | --- |
| 参数 | 类型 | 说明 |
| account | String | 挂单方账号 |
| sequence | Integer | 取消的单子号 |

返回：Transaction对象

### 传入密钥

方法：tx.setSecret(secret);

参数：

|  |  |  |
| --- | --- | --- |
| 参数 | 类型 | 说明 |
| secret | String | 井通钱包私钥 |

### 取消挂单

方法：tx.submit(callback);

参数：无

取消挂单完整例子：

var jlib = require('jingtum-lib');

var Remote = jlib.Remote;

var remote = new Remote({server: 'ws://ts5.jingtum.com:5020', local\_sign:true});

remote.connect(function (err, result) {

if (err) {

return console.log('err:', err);

}

var options = {account: 'jB7rxgh43ncbTX4WeMoeadiGMfmfqY2xLZ', sequence: 688};

var tx = remote.buildOfferCancelTx(options);

tx.setSecret('sn37nYrQ6KPJvTFmaBYokS3FjXUWd');

tx.submit(function (err, result) {

if (err) {

console.log('err:', err);

}

else if (result) {

console.log('res:', result);

}

});

});

返回结果：

{

engine\_result: 'tesSUCCESS',

engine\_result\_code: 0,

engine\_result\_message: 'The transaction was applied. Only final in a validated ledger.',

tx\_blob: '1200082200000000240000079A2F223CADE8201900000798684000000000002710732102FE64E0C20F0058F22F3742EDC15F49F318C04F88B130742C68BAF3B1C89FD1677446304402207233C0103866BE592885745BDCD7EB501A49B704EAE21DBF5F2EC3106252F12C022003DFB50DD42F0F82BA1532C977853929615690A7261986903301CF15611CEEA0811472F05993EBA9858291D364EBF6EEC3D851BD3792',

tx\_json:

{ Account: 'jB7rxgh43ncbTX4WeMoeadiGMfmfqY2xLZ',

Fee: '10000',

Flags: 0,

OfferSequence: 1944,

Sequence: 1946,

SigningPubKey: '02FE64E0C20F0058F22F3742EDC15F49F318C04F88B130742C68BAF3B1C89FD167',

Timestamp: 574402024,

TransactionType: 'OfferCancel',

TxnSignature: '304402207233C0103866BE592885745BDCD7EB501A49B704EAE21DBF5F2EC3106252F12C022003DFB50DD42F0F82BA1532C977853929615690A7261986903301CF15611CEEA0',

hash: '44BD366DF7BDE52C52FB37B7653564BF685B84B5E51A70E23B0BDCB26D4BA90F' }

}

返回结果说明：

|  |  |  |  |
| --- | --- | --- | --- |
| 参数 | | 类型 | 说明 |
| engine\_result | | String | 请求结果 |
| engine\_result\_code | | Array | 请求结果编码 |
| engine\_result\_message | | String | 请求结果message信息 |
| tx\_blob | | String | 16进制签名后的交易 |
| tx\_json | | Object | 交易内容 |
|  | Account | String | 账号地址 |
| Fee | String | 交易费 |
| Flags | Integer | 交易标记 |
| OfferSequence | Integer | 取消的单子号 |
| Sequence | Integer | 单子序列号 |
| SigningPubKey | String | 签名公钥 |
| Timestamp | Integer | 时间戳 |
| TransactionType | String | 交易类型：OfferCancel取消订单 |
| TxnSignature | String | 交易签名 |
| hash | String | 交易hash |

## 部署合约（Lua版）

首先通过deployContractTx方法返回一个Transaction对象，然后通过setSecret传入密钥，最后通过submit方法部署合约。

### 创建部署合约对象

方法：remote.deployContractTx({});

参数：

|  |  |  |
| --- | --- | --- |
| 参数 | 类型 | 说明 |
| account | String | 合约交易源账号 |
| amount | String/Number | 支付金额(最多支持六位小数) |
| payload | String | 智能合约代码(16进制字符串) |

可选参数：

|  |  |  |
| --- | --- | --- |
| 参数 | 类型 | 说明 |
| params | String | 合约参数 |

返回：Transaction对象

### 传入密钥

方法：tx.setSecret(secret);

参数：

|  |  |  |
| --- | --- | --- |
| 参数 | 类型 | 说明 |
| secret | String | 源账号私钥 |

### 部署合约

方法：tx.submit(callback);

参数：无

部署合约完整例子：

var jlib = require('jingtum-lib');

var Remote = jlib.Remote;

var utils = jlib.utils;

var remote = new Remote({server: 'ws://ts5.jingtum.com:5020', local\_sign:true});

remote.connect(function (err, result) {

if (err) {

return console.log('err:', err);

}

var options = {account: 'jHb9\*\*\*tyTh', amount: 10,payload: utils.stringToHex('result={}; function Init(t) result=scGetAccountBalance(t) return result end; function foo(t) result=scGetAccountBalance(t) return result end'),params: ['jHb9\*\*\*tyTh']};

var tx = remote.deployContractTx(options);

tx.setSecret('s\*\*');

tx.submit(function (err, result) {

if (err) {

console.log('err:', err);

}

else if (result) {

console.log('res:', result);

}

});

});

返回结果：

{

ContractState: 'jQUzfdE2ZnQLz3AbxuSbyyyQhuoXAxUn3A',

engine\_result: 'tesSUCCESS',

engine\_result\_code: 0,

engine\_result\_message: 'The transaction was applied. Only final in a validated ledger.',

tx\_blob

tx\_json:

{ Account: 'jHb9\*\*\*tyTh',

Amount: '10000000',

Args: [ [Object] ],

Fee: '10000',

Flags: 0,

Method: 0,

Payload: '726573756C743D7B7D3B202066756E6374696F6E20496E69742874292020726573756C743D73634765744163636F756E7442616C616E6365287429202072657475726E20726573756C742020656E643B202066756E6374696F6E20666F6F2874292020726573756C743D73634765744163636F756E7442616C616E6365287429202072657475726E20726573756C742020656E64',

Sequence: 1274,

SigningPubKey: '0330E7FC9D56BB25D6893BA3F317AE5BCF33B3291BD63DB32654A313222F7FD020',

TransactionType: 'ConfigContract',

TxnSignature: '3045022100E53D151ED6EEC46124CB75C0ED0DCACC7D4458B56109252431DAA80D11D1099802207FE4C2987AFBBD7F1FEC6DBFBD9A32C35D2F82CEC44E5AA6EE3DE0E58F1D8B62',

hash: 'D7E40A7164C11BFA81C05117010631EA5BCBD8A1A0B3B2FF343D9FB3F3575936' }

}

返回结果说明：

|  |  |  |  |
| --- | --- | --- | --- |
| 参数 | | 类型 | 说明 |
| ContractState | | String | 生成的合约地址 |
| engine\_result | | String | 请求结果 |
| engine\_result\_code | | Array | 请求结果编码 |
| engine\_result\_message | | String | 请求结果message信息 |
| tx\_blob | | String | 16进制签名后的交易 |
| tx\_json | | Object | 交易内容 |
|  | Account | String | 账号地址 |
| Fee | String | 交易费 |
| Flags | Integer | 交易标记 |
| Method | Integer | 合约交易方法：0表示部署；1表示调用 |
| Payload | Integer | 16进制合约代码 |
| Sequence | Integer | 单子序列号 |
| SigningPubKey | String | 签名公钥 |
| TransactionType | String | 交易类型：ConfigContract部署合约 |
| TxnSignature | String | 交易签名 |
| hash | String | 交易hash |

## 执行合约（Lua版）

首先通过callContractTx方法返回一个Transaction对象，然后通过setSecret传入密钥，最后通过submit方法执行合约。

### 创建执行合约对象

方法：remote.callContractTx({});

参数：

|  |  |  |
| --- | --- | --- |
| 参数 | 类型 | 说明 |
| account | String | 合约交易源账号 |
| destination | String | 合约地址 |
| func | String | 合约函数名 |

可选参数：

|  |  |  |
| --- | --- | --- |
| 参数 | 类型 | 说明 |
| params | String | 合约参数 |

返回：Transaction对象

### 传入密钥

方法：tx.setSecret(secret);

参数：

|  |  |  |
| --- | --- | --- |
| 参数 | 类型 | 说明 |
| secret | String | 源账号私钥 |

### 执行合约

方法：tx.submit(callback);

参数：无

部署合约完整例子：

var jlib = require('jingtum-lib');

var Remote = jlib.Remote;

var remote = new Remote({server: 'ws://ts5.jingtum.com:5020', local\_sign:true});

remote.connect(function (err, result) {

if (err) {

return console.log('err:', err);

}

var options = {account: 'jHb9\*\*\*tyTh', destination: 'j4YVQxCxaRRQ6gCVUvi9MoiTfWyPRnHwej',func: 'foo',params: ['jHb9\*\*\*tyTh']};

var tx = remote.callContractTx(options);

tx.setSecret('s\*\*');

tx.submit(function (err, result) {

if (err) {

console.log('err:', err);

}

else if (result) {

console.log('res:', result);

}

});

});

返回结果：

{

ContractState: '599868797812411271',

engine\_result: 'tesSUCCESS',

engine\_result\_code: 0,

engine\_result\_message: 'The transaction was applied. Only final in a validated ledger.',

tx\_blob: '12001E220000000024000004FC20240000000168400000000000271073210330E7FC9D56BB25D6893BA3F317AE5BCF33B3291BD63DB32654A313222F7FD02074473045022100835BEDFD6794F3B95A31971EDBF1865F9190D1A66FFC9515883B15107B368894022052F44E0E16B6A3BC3C8CF2A3881E35687060EB73B6B0503611CEF58781A512EA701103666F6F8114B5F762798A53D543A014CAF8B297CFF8F2F937E88314EC4F411C3B37C3E451D462C27489B0549650572CFAEB7012226A486239434A41577942346A7239315652576E3936446B756B473462776474795468E1F1',

tx\_json:

{ Account: 'jHb9\*\*\*tyTh',

Args: [ [Object] ],

ContractMethod: '666F6F',

Destination: 'j4YVQxCxaRRQ6gCVUvi9MoiTfWyPRnHwej',

Fee: '10000',

Flags: 0,

Method: 1,

Sequence: 1276,

SigningPubKey: '0330E7FC9D56BB25D6893BA3F317AE5BCF33B3291BD63DB32654A313222F7FD020',

TransactionType: 'ConfigContract',

TxnSignature: '3045022100835BEDFD6794F3B95A31971EDBF1865F9190D1A66FFC9515883B15107B368894022052F44E0E16B6A3BC3C8CF2A3881E35687060EB73B6B0503611CEF58781A512EA',

hash: '8DE51A01F6FA55F5FDBB196A39A1F4220772E19835C81CE2E8A44B487BCD961C' }

}

返回结果说明：

|  |  |  |  |
| --- | --- | --- | --- |
| 参数 | | 类型 | 说明 |
| ContractState | | String | 调用的合约结果 |
| engine\_result | | String | 请求结果 |
| engine\_result\_code | | Array | 请求结果编码 |
| engine\_result\_message | | String | 请求结果message信息 |
| tx\_blob | | String | 16进制签名后的交易 |
| tx\_json | | Object | 交易内容 |
|  | Account | String | 账号地址 |
| Args | Array | 合约传入的参数 |
| ContractMethod | String | 合约函数名 |
| Destination | String | 调用的合约地址 |
| Fee | String | 交易费 |
| Flags | Integer | 交易标记 |
| Method | Integer | 合约交易方法：0表示部署；1表示调用 |
| Sequence | Integer | 单子序列号 |
| SigningPubKey | String | 签名公钥 |
| TransactionType | String | 交易类型：ConfigContract合约类 |
| TxnSignature | String | 交易签名 |
| hash | String | 交易hash |

## 设置挂单佣金

首先通过buildBrokerageTx方法返回一个Transaction对象，然后通过setSecret传入密钥，最后通过submit方法设置平台手续费。

### 创建挂单佣金对象

方法：remote.buildBrokerageTx({});

参数：

|  |  |  |  |
| --- | --- | --- | --- |
| 参数 | | 类型 | 说明 |
| account | | String | 管理员账号 |
| feeAccount | | String | 收费账号 |
| mol | | Integer | 分子（0和正整数） |
| den | | Integer | 分母（正整数） |
| app | | Integer | 应用来源序号（正整数） |
| amount | | Object | 币种对象 |
|  | value | String | 数量，这里只是占位，没有实际意义。 |
| currency | String | 货币种类 |
| issuer | String | 货币发行方 |

返回：Transaction对象

### 传入密钥

方法：tx.setSecret(secret);

参数：

|  |  |  |
| --- | --- | --- |
| 参数 | 类型 | 说明 |
| secret | String | 管理员账号私钥 |

### 设置挂单佣金

方法：tx.submit(callback);

参数：无

设置挂单佣金完整例子：

var jlib = require('jingtum-lib');

var Remote = jlib.Remote;

var remote = new Remote({server: 'ws://ts5.jingtum.com:5020', local\_sign:true});

remote.connect(function (err, result) {

if (err) {

return console.log('err:', err);

}

var v = {

secret: 's...UTb',

address: 'j...yTh'

};

var req = remote.buildBrokerageTx({account: v.address, mol: 1, den: 1000, app:1,feeAccount:'jzTx4CRUZJT1ZsBhGHi7Wqikada63xRVv',

amount: {

"value": "3",

"currency": "TES",

"issuer": "jBciDE8Q3uJjf111VeiUNM775AMKHEbBLS"

}});

req.setSecret(v.secret);

tx.submit(function (err, result) {

if (err) {

console.log('err:', err);

}

else if (result) {

console.log('res:', result);

}

});

});

返回结果：

{

engine\_result: 'tesSUCCESS',

engine\_result\_code: 0,

engine\_result\_message: 'The transaction was applied. Only final in a validated ledger.',

tx\_blob: '1200CD220000000024000000012025000000013900000000000000013A00000000000003E861D38AA87BEE53800000000000000000000000005445535400000000007478E561645059399B334448F7544F2EF308ED3268400000000000271073210330E7FC9D56BB25D6893BA3F317AE5BCF33B3291BD63DB32654A313222F7FD02074473045022100FE513087425A863D1FB0004B29C6656E1B4C237F57F193FC5707317257926787022070AA7125CB5383F5B53000903BDAF9926CBE74F732DF7E09BE3EC581FEF5E7C08114B5F762798A53D543A014CAF8B297CFF8F2F937E8',

tx\_json:

{ Account: 'jHb9CJAWyB4jr91VRWn96DkukG4bwdtyTh',

Amount:

{ currency: 'TEST',

issuer: 'jBciDE8Q3uJjf111VeiUNM775AMKHEbBLS',

value: '3' },

AppType: 1,

Fee: '10000',

FeeAccountID: 'jzTx4CRUZJT1ZsBhGHi7Wqikada63xRVv',

Flags: 0,

OfferFeeRateDen: '00000000000003E8',

OfferFeeRateNum: '0000000000000001',

Sequence: 1,

SigningPubKey: '0330E7FC9D56BB25D6893BA3F317AE5BCF33B3291BD63DB32654A313222F7FD020',

TransactionType: 'Brokerage',

TxnSignature: '3045022100FE513087425A863D1FB0004B29C6656E1B4C237F57F193FC5707317257926787022070AA7125CB5383F5B53000903BDAF9926CBE74F732DF7E09BE3EC581FEF5E7C0',

hash: '3260743D0233EA5B8E2963F00D35B4198BA9EC6BE7638E84C9375A3CBE2172A3' }

}

返回结果说明：

|  |  |  |  |
| --- | --- | --- | --- |
| 参数 | | 类型 | 说明 |
| engine\_result | | String | 请求结果 |
| engine\_result\_code | | Array | 请求结果编码 |
| engine\_result\_message | | String | 请求结果message信息 |
| tx\_blob | | String | 16进制签名后的交易 |
| tx\_json | | Object | 交易内容 |
|  | Account | String | 管理员账号地址 |
| Amount | Object | 收交易手续费的币种信息 |
| AppType | Integer | 应用来源 |
| Fee | String | 网络费 |
| feeAccountID | String | 收费账号 |
| Flags | Integer | 交易标记 |
| OfferFeeRateDen | String | 分母 |
| OfferFeeRateNum | String | 分子 |
| Sequence | Integer | 单子序列号 |
| SigningPubKey | String | 签名公钥 |
| TransactionType | String | 交易类型：Brokerage设置交易手续费类 |
| TxnSignature | String | 交易签名 |
| hash | String | 交易hash |

## 部署合约（Solidity版）

首先通过initContract方法返回一个Transaction对象，然后通过setSecret传入密钥，最后通过submit方法完成合约的部署。

### 创建合约部署对象

方法：remote.initContract({});

参数：

|  |  |  |
| --- | --- | --- |
| 参数 | 类型 | 说明 |
| account | String | 合约发布者 |
| amount | Integer | 手续费 |
| payload | String | 合约编译后的16进制字节码 |
| abi | Array | 合约abi |
| params | Array | 可选，合约初始化参数 |

返回：Transaction对象

### 传入密钥

方法：tx.setSecret(secret);

参数：

|  |  |  |
| --- | --- | --- |
| 参数 | 类型 | 说明 |
| secret | String | 合约发布者账号私钥 |

### 部署合约

方法：tx.submit(callback);

参数：无

部署合约完整例子(solidity源码见文件最后[Solidity ERC20源码])：

var jlib = require('jingtum-lib');

var Remote = jlib.Remote;

var remote = new Remote({server: 'ws://ts5.jingtum.com:5020', local\_sign:true});

remote.connect(function (err, result) {

if (err) {

return console.log('err:', err);

}

var v = {

secret: 's...UTb',

address: 'j...yTh'

};

const abi = [

{

"constant": true,

"inputs": [],

"name": "name",

"outputs": [

{

"name": "",

"type": "string"

}

],

"payable": false,

"stateMutability": "view",

"type": "function"

},

{

"constant": true,

"inputs": [],

"name": "totalSupply",

"outputs": [

{

"name": "",

"type": "uint256"

}

],

"payable": false,

"stateMutability": "view",

"type": "function"

},

{

"constant": true,

"inputs": [],

"name": "decimals",

"outputs": [

{

"name": "",

"type": "uint8"

}

],

"payable": false,

"stateMutability": "view",

"type": "function"

},

{

"constant": true,

"inputs": [

{

"name": "",

"type": "address"

}

],

"name": "balanceOf",

"outputs": [

{

"name": "",

"type": "uint256"

}

],

"payable": false,

"stateMutability": "view",

"type": "function"

},

{

"constant": true,

"inputs": [],

"name": "symbol",

"outputs": [

{

"name": "",

"type": "string"

}

],

"payable": false,

"stateMutability": "view",

"type": "function"

},

{

"constant": false,

"inputs": [

{

"name": "\_to",

"type": "address"

},

{

"name": "\_value",

"type": "uint256"

}

],

"name": "transfer",

"outputs": [],

"payable": false,

"stateMutability": "nonpayable",

"type": "function"

},

{

"constant": true,

"inputs": [

{

"name": "",

"type": "address"

},

{

"name": "",

"type": "address"

}

],

"name": "allowance",

"outputs": [

{

"name": "",

"type": "uint256"

}

],

"payable": false,

"stateMutability": "view",

"type": "function"

},

{

"inputs": [

{

"name": "initialSupply",

"type": "uint256"

},

{

"name": "tokenName",

"type": "string"

},

{

"name": "tokenSymbol",

"type": "string"

}

],

"payable": false,

"stateMutability": "nonpayable",

"type": "constructor"

},

{

"payable": false,

"stateMutability": "nonpayable",

"type": "fallback"

}

];

var tx = remote.initContract({

account: v.address,

amount: 10,

payload: '',

abi: abi,

params:[2000, 'TestCurrency', 'TEST1']

});

tx.setSecret(v.secret);

tx.submit(function (err, result) {

if (err) {

console.log('err:', err);

}

else if (result) {

console.log('res:', result);

}

});

});

返回结果：

{

ContractState: 'jPZ1....9Kkh',

engine\_result: 'tesSUCCESS',

engine\_result\_code: 0,

engine\_result\_message: 'The transaction was applied. Only final in a validated ledger.',

tx\_blob: '',

tx\_json:

{ Account: 'jHb9....tyTh',

Amount: '10000000',

Fee: '10000',

Flags: 0,

Method: 0,

Payload: '36303630363034303532363031323630303236303030363130313030306138313534383136306666303231393136393038333630666631363032313739303535353033343135363130303262353736303030383066643562363034303531363130393237333830333830363130393237383333393831303136303430353238303830353139303630323030313930393139303830353138323031393139303630323030313830353138323031393139303530353036303032363030303930353439303631303130303061393030343630666631363630666631363630306130613833303236303033383139303535353036303033353436303034363030303333373366666666666666666666666666666666666666666666666666666666666666666666666666666666313637336666666666666666666666666666666666666666666666666666666666666666666666666666666631363831353236303230303139303831353236303230303136303030323038313930353535303831363030303930383035313930363032303031393036313030643339323931393036313030663335363562353038303630303139303830353139303630323030313930363130306561393239313930363130306633353635623530353035303530363130313938353635623832383035343630303138313630303131363135363130313030303230333136363030323930303439303630303035323630323036303030323039303630316630313630323039303034383130313932383236303166313036313031333435373830353136306666313931363833383030313137383535353631303136323536356238323830303136303031303138353535383231353631303136323537393138323031356238323831313131353631303136313537383235313832353539313630323030313931393036303031303139303631303134363536356235623530393035303631303136663931393036313031373335363562353039303536356236313031393539313930356238303832313131353631303139313537363030303831363030303930353535303630303130313631303137393536356235303930353635623930353635623631303738303830363130316137363030303339363030306633303036303630363034303532363030343336313036313030383335373630303033353763303130303030303030303030303030303030303030303030303030303030303030303030303030303030303030303030303030303030303030303930303436336666666666666666313638303633303666646465303331343631303039333537383036333138313630646464313436313031323135373830363333313363653536373134363130313461353738303633373061303832333131343631303137393537383036333935643839623431313436313031633635373830363361393035396362623134363130323534353738303633646436326564336531343631303239363537356233343135363130303865353736303030383066643562363030303830666435623334313536313030396535373630303038306664356236313030613636313033303235363562363034303531383038303630323030313832383130333832353238333831383135313831353236303230303139313530383035313930363032303031393038303833383336303030356238333831313031353631303065363537383038323031353138313834303135323630323038313031393035303631303063623536356235303530353035303930353039303831303139303630316631363830313536313031313335373830383230333830353136303031383336303230303336313031303030613033313931363831353236303230303139313530356235303932353035303530363034303531383039313033393066333562333431353631303132633537363030303830666435623631303133343631303361303536356236303430353138303832383135323630323030313931353035303630343035313830393130333930663335623334313536313031353535373630303038306664356236313031356436313033613635363562363034303531383038323630666631363630666631363831353236303230303139313530353036303430353138303931303339306633356233343135363130313834353736303030383066643562363130316230363030343830383033353733666666666666666666666666666666666666666666666666666666666666666666666666666666663136393036303230303139303931393035303530363130336239353635623630343035313830383238313532363032303031393135303530363034303531383039313033393066333562333431353631303164313537363030303830666435623631303164393631303364313536356236303430353138303830363032303031383238313033383235323833383138313531383135323630323030313931353038303531393036303230303139303830383338333630303035623833383131303135363130323139353738303832303135313831383430313532363032303831303139303530363130316665353635623530353035303530393035303930383130313930363031663136383031353631303234363537383038323033383035313630303138333630323030333631303130303061303331393136383135323630323030313931353035623530393235303530353036303430353138303931303339306633356233343135363130323566353736303030383066643562363130323934363030343830383033353733666666666666666666666666666666666666666666666666666666666666666666666666666666663136393036303230303139303931393038303335393036303230303139303931393035303530363130343666353635623030356233343135363130326131353736303030383066643562363130326563363030343830383033353733666666666666666666666666666666666666666666666666666666666666666666666666666666663136393036303230303139303931393038303335373366666666666666666666666666666666666666666666666666666666666666666666666666666666313639303630323030313930393139303530353036313034376535363562363034303531383038323831353236303230303139313530353036303430353138303931303339306633356236303030383035343630303138313630303131363135363130313030303230333136363030323930303438303630316630313630323038303931303430323630323030313630343035313930383130313630343035323830393239313930383138313532363032303031383238303534363030313831363030313136313536313031303030323033313636303032393030343830313536313033393835373830363031663130363130333664353736313031303038303833353430343032383335323931363032303031393136313033393835363562383230313931393036303030353236303230363030303230393035623831353438313532393036303031303139303630323030313830383331313631303337623537383239303033363031663136383230313931356235303530353035303530383135363562363030333534383135363562363030323630303039303534393036313031303030613930303436306666313638313536356236303034363032303532383036303030353236303430363030303230363030303931353039303530353438313536356236303031383035343630303138313630303131363135363130313030303230333136363030323930303438303630316630313630323038303931303430323630323030313630343035313930383130313630343035323830393239313930383138313532363032303031383238303534363030313831363030313136313536313031303030323033313636303032393030343830313536313034363735373830363031663130363130343363353736313031303038303833353430343032383335323931363032303031393136313034363735363562383230313931393036303030353236303230363030303230393035623831353438313532393036303031303139303630323030313830383331313631303434613537383239303033363031663136383230313931356235303530353035303530383135363562363130343761333338333833363130346133353635623530353035363562363030353630323035323831363030303532363034303630303032303630323035323830363030303532363034303630303032303630303039313530393135303530353438313536356236303030383038333733666666666666666666666666666666666666666666666666666666666666666666666666666666663136313431353135313536313034636135373630303038306664356238313630303436303030383637336666666666666666666666666666666666666666666666666666666666666666666666666666666631363733666666666666666666666666666666666666666666666666666666666666666666666666666666663136383135323630323030313930383135323630323030313630303032303534313031353135313536313035313835373630303038306664356236303034363030303834373366666666666666666666666666666666666666666666666666666666666666666666666666666666313637336666666666666666666666666666666666666666666666666666666666666666666666666666666631363831353236303230303139303831353236303230303136303030323035343832363030343630303038363733666666666666666666666666666666666666666666666666666666666666666666666666666666663136373366666666666666666666666666666666666666666666666666666666666666666666666666666666313638313532363032303031393038313532363032303031363030303230353430313131313531353631303561363537363030303830666435623630303436303030383437336666666666666666666666666666666666666666666666666666666666666666666666666666666631363733666666666666666666666666666666666666666666666666666666666666666666666666666666663136383135323630323030313930383135323630323030313630303032303534363030343630303038363733666666666666666666666666666666666666666666666666666666666666666666666666666666663136373366666666666666666666666666666666666666666666666666666666666666666666666666666666313638313532363032303031393038313532363032303031363030303230353430313930353038313630303436303030383637336666666666666666666666666666666666666666666666666666666666666666666666666666666631363733666666666666666666666666666666666666666666666666666666666666666666666666666666663136383135323630323030313930383135323630323030313630303032303630303038323832353430333932353035303831393035353530383136303034363030303835373366666666666666666666666666666666666666666666666666666666666666666666666666666666313637336666666666666666666666666666666666666666666666666666666666666666666666666666666631363831353236303230303139303831353236303230303136303030323036303030383238323534303139323530353038313930353535303830363030343630303038353733666666666666666666666666666666666666666666666666666666666666666666666666666666663136373366666666666666666666666666666666666666666666666666666666666666666666666666666666313638313532363032303031393038313532363032303031363030303230353436303034363030303837373366666666666666666666666666666666666666666666666666666666666666666666666666666666313637336666666666666666666666666666666666666666666666666666666666666666666666666666666631363831353236303230303139303831353236303230303136303030323035343031313431353135363130373465353766653562353035303530353035363030613136353632376137613732333035383230623236363564663064386438353232383033613139616336626339386666303130313231653131633136643033343265616365643031643934313030636531383030323930303030303030303030303030303030303030303030303030303030303030303030303030303030303030303030303030303030303030303030303030376430303030303030303030303030303030303030303030303030303030303030303030303030303030303030303030303030303030303030303030303030303036303030303030303030303030303030303030303030303030303030303030303030303030303030303030303030303030303030303030303030303030303030613030303030303030303030303030303030303030303030303030303030303030303030303030303030303030303030303030303030303030303030303030303063353436353733373434333735373237323635366536333739303030303030303030303030303030303030303030303030303030303030303030303030303030303030303030303030303030303030303030303030303030303030303030303030303030303030303030303030303030303030303030303030303030303030303535343435353335343331303030303030303030303030303030303030303030303030303030303030303030303030303030303030303030303030303030303030',

Sequence: 413,

SigningPubKey: '0330E7FC9D56BB25D6893BA3F317AE5BCF33B3291BD63DB32654A313222F7FD020',

TransactionType: 'AlethContract',

TxnSignature: '304402206C941CF3355094C5E85C2410FBB131288802094B790C399639A543A9DD64DDB1022059EEA991A946A28EA9695AEB2F5F724831C371BB0F9C72D50DD6855BCDF5C07D',

hash: '7BAAF8543E5E5761500CB21F5BCBCB9CA7A786704E82153481A7944E8F6E97F9'

}

返回结果说明：

|  |  |  |  |
| --- | --- | --- | --- |
| 参数 | | 类型 | 说明 |
| ContractState | | String | 返回合约账号 |
| engine\_result | | String | 请求结果 |
| engine\_result\_code | | Array | 请求结果编码 |
| engine\_result\_message | | String | 请求结果message信息 |
| tx\_blob | | String | 16进制签名后的交易 |
| tx\_json | | Object | 交易内容 |
|  | Account | String | 合约发起者账号地址 |
| Amount | Object | 收交易手续费的币种信息 |
| Fee | String | 网络费 |
| Flags | Integer | 交易标记 |
| Method | Integer | 合约方法：0表示部署，1表示调用 |
| Payload | String | 合约编译后的16进制字节码 |
| Sequence | Integer | 单子序列号 |
| SigningPubKey | String | 签名公钥 |
| TransactionType | String | 交易类型：AlethContract合约类 |
| TxnSignature | String | 交易签名 |
| hash | String | 交易hash |

## 调用合约（Solidity版）

首先通过invokeContract方法返回一个Transaction对象，然后通过setSecret传入密钥，最后通过submit方法完成合约的调用。

### 创建合约调用对象

方法：remote.invokeContract({});

参数：

|  |  |  |
| --- | --- | --- |
| 参数 | 类型 | 说明 |
| account | String | 合约调用者 |
| destination | String | 合约账号 |
| abi | Array | 合约abi |
| func | String | 合约函数名及参数 |
| amount | Number | 可选，合约中msg.value的值。只有payable修饰的合约函数，才可以设置该值，默认为0。 |

返回：Transaction对象

### 传入密钥

方法：tx.setSecret(secret);

参数：

|  |  |  |
| --- | --- | --- |
| 参数 | 类型 | 说明 |
| secret | String | 合约调用者账号私钥 |

### 调用合约

方法：tx.submit(callback);

参数：无

调用合约完整例子(solidity源码见文件最后[Solidity ERC20源码])：

var jlib = require('jingtum-lib');

var Remote = jlib.Remote;

var remote = new Remote({server: 'ws://ts5.jingtum.com:5020', local\_sign:true});

remote.connect(function (err, result) {

if (err) {

return console.log('err:', err);

}

var v = {

secret: 's...UTb',

address: 'j...yTh'

};

const abi = [

{

"constant": true,

"inputs": [],

"name": "name",

"outputs": [

{

"name": "",

"type": "string"

}

],

"payable": false,

"stateMutability": "view",

"type": "function"

},

{

"constant": true,

"inputs": [],

"name": "totalSupply",

"outputs": [

{

"name": "",

"type": "uint256"

}

],

"payable": false,

"stateMutability": "view",

"type": "function"

},

{

"constant": true,

"inputs": [],

"name": "decimals",

"outputs": [

{

"name": "",

"type": "uint8"

}

],

"payable": false,

"stateMutability": "view",

"type": "function"

},

{

"constant": true,

"inputs": [

{

"name": "",

"type": "address"

}

],

"name": "balanceOf",

"outputs": [

{

"name": "",

"type": "uint256"

}

],

"payable": false,

"stateMutability": "view",

"type": "function"

},

{

"constant": true,

"inputs": [],

"name": "symbol",

"outputs": [

{

"name": "",

"type": "string"

}

],

"payable": false,

"stateMutability": "view",

"type": "function"

},

{

"constant": false,

"inputs": [

{

"name": "\_to",

"type": "address"

},

{

"name": "\_value",

"type": "uint256"

}

],

"name": "transfer",

"outputs": [],

"payable": false,

"stateMutability": "nonpayable",

"type": "function"

},

{

"constant": true,

"inputs": [

{

"name": "",

"type": "address"

},

{

"name": "",

"type": "address"

}

],

"name": "allowance",

"outputs": [

{

"name": "",

"type": "uint256"

}

],

"payable": false,

"stateMutability": "view",

"type": "function"

},

{

"inputs": [

{

"name": "initialSupply",

"type": "uint256"

},

{

"name": "tokenName",

"type": "string"

},

{

"name": "tokenSymbol",

"type": "string"

}

],

"payable": false,

"stateMutability": "nonpayable",

"type": "constructor"

},

{

"payable": false,

"stateMutability": "nonpayable",

"type": "fallback"

}

];

var tx = remote.invokeContract({

account: v.address,

destination: 'jPZ1....9Kkh',

abi: abi,

func:"transfer('jPZ1....9Kkh', 15)"});

tx.setSecret(v.secret);

tx.submit(function (err, result) {

if (err) {

console.log('err:', err);

}

else if (result) {

console.log('res:', result);

}

});

});

返回结果：

{

ContractState: '',

engine\_result: 'tesSUCCESS',

engine\_result\_code: 0,

engine\_result\_message: 'The transaction was applied. Only final in a validated ledger.',

tx\_blob

tx\_json:

{ Account: 'jHb9CJAWyB4jr91VRWn96DkukG4bwdtyTh',

Args: [ { Arg:

{ ContractParamsType: 0,

Parameter: '61393035396362623030303030303030303030303030303030303030303030306637386136396463643533303864623330633431373234663034656638366361336163373461623330303030303030303030303030303030303030303030303030303030303030303030303030303030303030303030303030303030303030303030303030303066' } } ],

Destination: 'jPZ1....9Kkh',

Fee: '10000',

Flags: 0,

Method: 1,

Sequence: 414,

SigningPubKey: '0330E7FC9D56BB25D6893BA3F317AE5BCF33B3291BD63DB32654A313222F7FD020',

TransactionType: 'AlethContract',

TxnSignature: '3045022100C44640B21BF99C1764042A87FCAE6877BF1188F9F8DD887969EEF735333E584002204A2158F843021522FF7F93940BA4C1A933E21F03B6B9E5A3F3A54C217428CDDC',

hash: '561190424A8325FE6A93A2E3D1E16852B67CCC46354DA766F63C60A73403377A'

}

返回结果说明：

|  |  |  |  |
| --- | --- | --- | --- |
| 参数 | | 类型 | 说明 |
| ContractState | | String | 调用的合约方法返回值，没有返回值显示空字符串 |
| engine\_result | | String | 请求结果 |
| engine\_result\_code | | Array | 请求结果编码 |
| engine\_result\_message | | String | 请求结果message信息 |
| tx\_blob | | String | 16进制签名后的交易 |
| tx\_json | | Object | 交易内容 |
|  | Account | String | 合约发起者账号地址 |
| Args | Array | 收交易手续费的币种信息 |
| Destination | String | 合约账号 |
| Fee | String | 网络费 |
| Flags | Integer | 交易标记 |
| Method | Integer | 合约方法：0表示部署，1表示调用 |
| Sequence | Integer | 单子序列号 |
| SigningPubKey | String | 签名公钥 |
| TransactionType | String | 交易类型：AlethContract合约类 |
| TxnSignature | String | 交易签名 |
| hash | String | 交易hash |

## 监听事件

Remote有两个监听事件：监听所有交易(transactions)和监听所有账本(ledger\_closed)，监听结果放到回调函数中，回调中只有一个参数，为监听到的消息。

方法：remote.on('transactions',callback);

方法：remote.on('ledger\_closed',callback);

例子：

var jlib = require('jingtum-lib');

var Remote = jlib.Remote;

var remote = new Remote({server: 'ws://ts5.jingtum.com:5020', local\_sign:true});

remote.connect(function (err, result) {

if (err) {

return console.log('err:', err);

}

remote.on('transactions', function (msg) {

console.log('tx: ',msg);

});

remote.on('ledger\_closed', function (msg) {

console.log('ledger: ',msg);

});

});

# Request类

Request类主管GET请求，包括获得服务器、账号、挂单、路径等信息。请求时不需要提供密钥，且对所有用户公开。所有的请求是异步的，会提供一个回调函数。每个回调函数有两个参数，一个是错误，另一个是结果。提供以下方法：

\* selectLedger(ledger)

\* submit(callback)

## 指定账本

方法：selectLedger(ledger);

参数：

|  |  |  |
| --- | --- | --- |
| 参数 | 类型 | 说明 |
| ledger | String | 账本高度或者账号hash |

例子：

var jlib = require('jingtum-lib');

var Remote = jlib.Remote;

var remote = new Remote({server: 'ws://ts5.jingtum.com:5020', local\_sign:true});

remote.connect(function (err, result) {

if (err) {

return console.log('err:', err);

}

var req = remote.requestAccountInfo({account: 'jB7rxgh43ncbTX4WeMoeadiGMfmfqY2xLZ'});

req.selectLedger("8573498");

req.submit(function(err, result) {

if(err) {console.log('err:',err);}

else if(result){

console.log('res:', result);

}

});

});

## 提交请求

方法：submit(callback);

参数：回调函数，包含两个参数：错误信息和结果信息

例子：

var jlib = require('jingtum-lib');

var Remote = jlib.Remote;

var remote = new Remote({server: 'ws://ts5.jingtum.com:5020', local\_sign:true});

remote.connect(function (err, result) {

if (err) {

return console.log('err:', err);

}

var req = remote.requestAccountInfo({account: 'jB7rxgh43ncbTX4WeMoeadiGMfmfqY2xLZ'});

req.submit(function(err, result) {

if(err) {console.log('err:',err);}

else if(result){

console.log('res:', result);

}

});

});

# Transaction类

Transaction类主管POST请求，包括组装交易和交易参数。请求时需要提供密钥，且交易可以进行本地签名和服务器签名。目前支持服务器签名，本地签名支持主要的交易，还有部分参数不支持。所有的请求是异步的，会提供一个回调函数。每个回调函数有两个参数，一个是错误，另一个是结果。提供以下方法：

\* getAccount()

\* getTransactionType()

\* setSecret(secret)

\* addMemo(memo)

\* setPath(key)

\* setSendMax(amount)

\* setTransferRate(rate)

\* setFlags(flags)

\* submit(callback)

## 获得交易账号

方法：getAccount();

参数：无

返回：账号

例子：

var jlib = require('jingtum-lib');

var Remote = jlib.Remote;

var remote = new Remote({server: 'ws://ts5.jingtum.com:5020'});

var options = {account: 'jB7rxgh43ncbTX4WeMoeadiGMfmfqY2xLZ', sequence: 688};

var tx = remote.buildOfferCancelTx(options);

var account = tx.getAccount();

console.log(account);

## 获得交易类型

方法：getTransactionType();

参数：无

返回：交易类型

例子：

var jlib = require('jingtum-lib');

var Remote = jlib.Remote;

var remote = new Remote({server: 'ws://ts5.jingtum.com:5020'});

var options = {account: 'jB7rxgh43ncbTX4WeMoeadiGMfmfqY2xLZ', sequence: 688};

var tx = remote.buildOfferCancelTx(options);

var type = tx.getTransactionType();

console.log(type);

## 传入私钥

交易提交之前需要传入私钥。

方法：setSecret(secret);

参数：

|  |  |  |
| --- | --- | --- |
| 参数 | 类型 | 说明 |
| secret | String | 井通钱包私钥 |

返回：Transaction对象

例子：

var jlib = require('jingtum-lib');

var Remote = jlib.Remote;

var remote = new Remote({server: 'ws://ts5.jingtum.com:5020'});

var options = {account: 'jB7rxgh43ncbTX4WeMoeadiGMfmfqY2xLZ', sequence: 688};

var tx = remote.buildOfferCancelTx(options);

tx.setSecret('sn37nYrQ6KPJvTFmaBYokS3FjXUWd');;

## 添加备注

方法：addMemo(memo);

参数：

|  |  |  |
| --- | --- | --- |
| 参数 | 类型 | 说明 |
| memo | String | 备注信息，不超过2k。 |

返回：Transaction对象

例子：

var jlib = require('jingtum-lib');

var Remote = jlib.Remote;

var remote = new Remote({server: 'ws://ts5.jingtum.com:5020'});

var tx = remote.buildPaymentTx({

account: 'jB7rxgh43ncbTX4WeMoeadiGMfmfqY2xLZ',

to: 'jDUjqoDZLhzx4DCf6pvSivjkjgtRESY62c',

amount: {

"value": 0.5,

"currency": "SWT",

"issuer": ""

}

});

tx.addMemo('给jDUjqoDZLhzx4DCf6pvSivjkjgtRESY62c支付0.5swt.');

## 提交请求

方法：submit(callback);

参数：回调函数，包含两个参数：错误信息和结果信息

例子：

var jlib = require('jingtum-lib');

var Remote = jlib.Remote;

var remote = new Remote({server: 'ws://ts5.jingtum.com:5020', local\_sign:true});

remote.connect(function (err, result) {

if (err) {

return console.log('err:', err);

}

var req = remote.requestAccountInfo({account: 'jB7rxgh43ncbTX4WeMoeadiGMfmfqY2xLZ'});

req.submit(function(err, result) {

if(err) {console.log('err:',err);}

else if(result){

console.log('res:', result);

}

});

});

# 工具类

Utils类是工具类，提供以下方法：

\* hexToString(hex)

\* stringToHex(str)

\* isValidAmount()

\* isValidAmount0()

\* parseAmount()

\* isValidCurrency()

\* isValidHash()

\* isValidAddress()

\* isValidSecret()

\* processTx()

## 16进制转字符串

方法：hexToString(hex)

参数：

|  |  |  |
| --- | --- | --- |
| 参数 | 类型 | 说明 |
| hex | String | 16进制字符串 |

返回：字符串

例子：

var jlib = require('jingtum-lib');

var utils= jlib.utils;

console.log(utils.hexToString('6e6968616f'));

返回结果：

nihao

## 字符串转16进制

方法：stringToHex(str)

参数：

|  |  |  |
| --- | --- | --- |
| 参数 | 类型 | 说明 |
| str | String | 字符串 |

返回：16进制字符串

例子：

var jlib = require('jingtum-lib');

var utils= jlib.utils;

console.log(utils.stringToHex(nihao));

返回结果：

6e6968616f

## 判断amount是否有效

方法：isValidAmount({})

参数：

|  |  |  |
| --- | --- | --- |
| 参数 | 类型 | 说明 |
| value | String | 数值 |
| currency | String | 货币种类 |
| issuer | String | 货币发行方 |

返回：Boolean值

例子：

var jlib = require('jingtum-lib');

var utils= jlib.utils;

console.log(utils.isValidAmount({value: '20', currency:'SWT', issuer:''}));

返回结果：

true

## 判断市场挂单的货币对是否有效

方法：isValidAmount0({})

参数：

|  |  |  |
| --- | --- | --- |
| 参数 | 类型 | 说明 |
| currency | String | 货币种类 |
| issuer | String | 货币发行方 |

返回：Boolean值

例子：

var jlib = require('jingtum-lib');

var utils= jlib.utils;

console.log(utils.isValidAmount0({currency:'SWT', issuer:''}));

返回结果：

true

## 统一基础货币SWT的amount格式

方法：parseAmount()

参数：数值/对象；

返回：若为数值，则默认转换成基础货币的amount格式（即{value: ‘数值’,currency:’SWT’,issuer:’’}）返回，**这里value值需乘以1000000**；若为合法amount对象，返回此对象；否则，返回null。

例子：

var jlib = require('jingtum-lib');

var utils= jlib.utils;

console.log(utils.parseAmount('2000000'));

返回结果：

{ value: '2', currency: 'SWT', issuer: '' }

## 判断currency是否合法

方法：isValidCurrency(currency)

参数：

|  |  |  |
| --- | --- | --- |
| 参数 | 类型 | 说明 |
| currency | String | 货币种类 |

注：currency为3-6位字母(不区分大小写，统一转换成大写)或者由A-F0-9组成的40位16进制字符串。

返回：Boolean值

例子：

var jlib = require('jingtum-lib');

var utils= jlib.utils;

console.log(utils.isValidCurrency('0CFDA38CC98A630F97B9FFFFB324FC40F2882F97'));

返回结果：

true

## 判断hash是否合法

方法：isValidHash(hash)

参数：

|  |  |  |
| --- | --- | --- |
| 参数 | 类型 | 说明 |
| hash | String | 交易hash/账本hash |

返回：Boolean值

例子：

var jlib = require('jingtum-lib');

var utils= jlib.utils;

console.log(utils.isValidHash('A89A245D311FCCF3C329E865AFF69C59AB9CD66F0808BD0C16D5706103319125'));

返回结果：

true

## 判断address是否合法

方法：isValidAddress(address)

参数：

|  |  |  |
| --- | --- | --- |
| 参数 | 类型 | 说明 |
| address | String | 钱包地址 |

注：井通地址以j开头。

返回：Boolean值

例子：

var jlib = require('jingtum-lib');

var utils= jlib.utils;

console.log(utils.isValidAddress('jsEtk7VtqY8PJ4uSmTbQ1KHkv1u9zhr8pQ'));

返回结果：

true

## 判断secret是否合法

方法：isValidSecret(secret)

参数：

|  |  |  |
| --- | --- | --- |
| 参数 | 类型 | 说明 |
| secret | String | 钱包私钥 |

注：井通私钥以s开头。

返回：Boolean值

例子：

var jlib = require('jingtum-lib');

var utils= jlib.utils;

console.log(utils.isValidSecret('sn5qWH2E18HbnuvfrHeMcnFdENcSu'));

返回结果：

true

## 从账号角度过滤交易信息

方法：processTx(tx, account)

参数：

|  |  |  |
| --- | --- | --- |
| 参数 | 类型 | 说明 |
| tx | Object | 底层返回的交易详情 |
| account | String | 钱包地址 |

注：井通私钥以s开头。

返回：Boolean值

例子：

var jlib = require('jingtum-lib');

var utils= jlib.utils;

var tx = { Account: 'jKCQAZwwN2sQG3Mb56GmWVqxkgpLwwAZuR',

Fee: '10000',

Flags: 524288,

Sequence: 650,

SigningPubKey: '03E791056E6B4C62E26C0F1F3BB89317667AB74170B49339972716FC53FFCF007C',

TakerGets: '2000000000',

TakerPays:

{ currency: 'CNY',

issuer: 'jGa9J9TkqtBcUoHe2zqhVFFbgUVED6o9or',

value: '13.46' },

Timestamp: 611552863,

TransactionType: 'OfferCreate',

TxnSignature: '3045022100B342C7159E1AD7FAA13452C1FB01F77A107263AFB06E93F9B4F307EF9DF9F98E0220690C964D89146250E879B9EA5ED311C3317631F1F6450360603E6818AFFB5FAF',

date: 611552870,

hash: 'AB2A25557FF03911A8FC0A412293BE9D9FCB20CDD530EE05957A9859F8467C32',

inLedger: 12839308,

ledger\_index: 12839308,

meta:

{ AffectedNodes: [ { ModifiedNode:

{ FinalFields: { Account: 'j9x4pABowsWxmK1DhhWyK34u3boC6h3LHe',

BookDirectory: '51603377F758E3C8FA007C77312DDA06A737A1395CD5FC435D0547675A0517F6',

BookNode: '0000000000000000',

Flags: 0,

OwnerNode: '0000000000000000',

Sequence: 7031,

TakerGets:

{ currency: 'CNY',

issuer: 'jGa9J9TkqtBcUoHe2zqhVFFbgUVED6o9or',

value: '1148.95233' },

TakerPays: '170721000000' },

LedgerEntryType: 'Offer',

LedgerIndex: '020110B8BED1F151B9D3AF9E5D412D8627CB08232B388ADE1F4B0C68E7608BEC',

PreviousFields: { TakerGets:

{ currency: 'CNY',

issuer: 'jGa9J9TkqtBcUoHe2zqhVFFbgUVED6o9or',

value: '1162.41233' },

TakerPays: '172721000000' },

PreviousTxnID: '9CB6AEFA273C750242D5B8AF4299347E77F9A47C5D0B89EE5F6A4D5577E8C4A0',

PreviousTxnLgrSeq: 12839301 } },

{ ModifiedNode:

{ FinalFields: { Account: 'jEoSyfChhUMzpRDttAJXuie8XhqyoPBYvV',

Balance: '533983297806',

Flags: 0,

OwnerCount: 1,

Sequence: 34380818 },

LedgerEntryType: 'AccountRoot',

LedgerIndex: '109E80FB8CC6D82D4F7F7D77248C2C3C116ECCD4520B3D2A88421FFF94A57B1E',

PreviousFields: { Balance: '533983287806', Sequence: 34380817 },

PreviousTxnID: '756338B8F9D4DCC8D88382B1092B13F75F65F330970278AFC7449496FF9875E9',

PreviousTxnLgrSeq: 12839308 } },

{ ModifiedNode:

{ FinalFields: { Balance:

{ currency: 'CNY',

issuer: 'jjjjjjjjjjjjjjjjjjjjBZbvri',

value: '-6872.222452374449' },

Flags: 2228224,

HighLimit:

{ currency: 'CNY',

issuer: 'jKCQAZwwN2sQG3Mb56GmWVqxkgpLwwAZuR',

value: '10000000000' },

HighNode: '0000000000000000',

LowLimit:

{ currency: 'CNY',

issuer: 'jGa9J9TkqtBcUoHe2zqhVFFbgUVED6o9or',

value: '0' },

LowNode: '00000000000012A0' },

LedgerEntryType: 'SkywellState',

LedgerIndex: '2600F8FCB87FEA15F74B0DB785016384C79AEA0730B62F597C1E576801BB813B',

PreviousFields: { Balance:

{ currency: 'CNY',

issuer: 'jjjjjjjjjjjjjjjjjjjjBZbvri',

value: '-6858.762452374449' } },

PreviousTxnID: '9B28F7958E729F0F904410B132D1F81481B38DD9F017790A82168CD38C995331',

PreviousTxnLgrSeq: 12838251 } },

{ ModifiedNode:

{ FinalFields:{ Account: 'j9x4pABowsWxmK1DhhWyK34u3boC6h3LHe',

Balance: '1496144192938',

Flags: 0,

OwnerCount: 8,

Sequence: 7032 },

LedgerEntryType: 'AccountRoot',

LedgerIndex: '40A20BDD3C226C987579F6C821BF84492E1C6B6EFB62311481BA6B8CB1D7775A',

PreviousFields: { Balance: '1494144192938' },

PreviousTxnID: '9CB6AEFA273C750242D5B8AF4299347E77F9A47C5D0B89EE5F6A4D5577E8C4A0',

PreviousTxnLgrSeq: 12839301 } },

{ ModifiedNode:

{ FinalFields: { Account: 'jKCQAZwwN2sQG3Mb56GmWVqxkgpLwwAZuR',

Balance: '500538133',

Flags: 0,

OwnerCount: 10,

Sequence: 651 },

LedgerEntryType: 'AccountRoot',

LedgerIndex: 'B39BD926378886F7EF4F81CEF862FC4D1E8E6D1265945AA9EC40FD85132DC629',

PreviousFields: { Balance: '2500548133', Sequence: 650 },

PreviousTxnID: '5BA24DE17EF64EDF942D99F247ED1495F5A61ED9260513FEDCA3E4BADBADFF3E',

PreviousTxnLgrSeq: 12839303 } },

{ ModifiedNode:

{ FinalFields: { Balance:

{ currency: 'CNY',

issuer: 'jjjjjjjjjjjjjjjjjjjjBZbvri',

value: '1148.954817858577' },

Flags: 1114112,

HighLimit:

{ currency: 'CNY',

issuer: 'jGa9J9TkqtBcUoHe2zqhVFFbgUVED6o9or',

value: '0' },

HighNode: '000000000000172A',

LowLimit:

{ currency: 'CNY',

issuer: 'j9x4pABowsWxmK1DhhWyK34u3boC6h3LHe',

value: '10000000000' },

LowNode: '0000000000000000' },

LedgerEntryType: 'SkywellState',

LedgerIndex: 'E3E9FE1827E83B52F7017D3038F8C769F09343801BB073A993DE620756069137',

PreviousFields: { Balance:

{ currency: 'CNY',

issuer: 'jjjjjjjjjjjjjjjjjjjjBZbvri',

value: '1162.414817858577' } },

PreviousTxnID: '9CB6AEFA273C750242D5B8AF4299347E77F9A47C5D0B89EE5F6A4D5577E8C4A0',

PreviousTxnLgrSeq: 12839301 } } ],

TransactionIndex: 3,

TransactionResult: 'tesSUCCESS' },

validated: true };

var newTx = utils.processTx(tx, tx.Account);

console.log(newTx);

返回结果：

{ date: 1558237670,

hash: 'AB2A25557FF03911A8FC0A412293BE9D9FCB20CDD530EE05957A9859F8467C32',

type: 'offernew',

fee: '0.01',

result: 'tesSUCCESS',

memos: [],

offertype: 'sell',

gets: { value: '2000', currency: 'SWT', issuer: '' },

pays:

{ currency: 'CNY',

issuer: 'jGa9J9TkqtBcUoHe2zqhVFFbgUVED6o9or',

value: '13.46' },

seq: 650,

price: 0.00673,

effects:

[ { effect: 'offer\_bought',

counterparty:

{ account: 'j9x4pABowsWxmK1DhhWyK34u3boC6h3LHe',

seq: 7031,

hash: '9CB6AEFA273C750242D5B8AF4299347E77F9A47C5D0B89EE5F6A4D5577E8C4A0' },

paid: { value: '2000', currency: 'SWT', issuer: '' },

got:

{ value: '13.46',

currency: 'CNY',

issuer: 'jGa9J9TkqtBcUoHe2zqhVFFbgUVED6o9or' },

type: 'sold',

price: '0.00673' } ],

balances: { CNY: 6872.222452374449, SWT: 500.538133 },

dealGets: { value: '2000', currency: 'SWT', issuer: '' },

dealPays:

{ value: '13.46',

currency: 'CNY',

issuer: 'jGa9J9TkqtBcUoHe2zqhVFFbgUVED6o9or' },

dealPrice: '0.00673',

dealNum: 2 }

返回结果说明：

|  |  |  |  |
| --- | --- | --- | --- |
| 参数 | | 类型 | 说明 |
| date | | Integer | 交易时间 |
| hash | | String | 交易hash |
| type | | String | 交易类型 |
| fee | | String | 燃料费 |
| result | | String | 交易结果码，tesSUCCESS表示成功，其他详见底层常见错误附录 |
| memos | | Array | 交易备注 |
| offertype | | String | 挂单买卖类型，只有挂单类才有此字段 |
| gets | | Object | 委托的对方将要获得的数量，只有挂单类才有此字段 |
|  | value | String | 数值 |
| currency | String | 货币 |
| issuer | String | 发行方 |
| pays | | Object | 委托的对方将要支付的数量，只有挂单类才有此字段 |
|  | value | String | 数值 |
| currency | String | 货币 |
| issuer | String | 发行方 |
| seq | | Integer | 该笔交易对应的该账号的sequence编号 |
| effects | | Array | 交易效果，包括创建挂单、取消挂单、挂单成交详情等，详见后面effects说明 |
| balances | | Object | 变动币种余额 |
| dealGets | | Object | 实际对方获得的，挂单成交才有此字段 |
|  | value | String | 金额 |
| currency | String | 币种 |
| issuer | String | 发行方 |
| dealPays | | Object | 实际对方支付的，挂单成交才有此字段 |
|  | value | String | 金额 |
| currency | String | 币种 |
| issuer | String | 发行方 |
| totalRate | | Object | 一共收取的挂单手续费，挂单成交才有此字段 |
|  | value | String | 金额 |
| currency | String | 币种 |
| issuer | String | 发行方 |
| dealPrice | | String | 实际成交价，挂单成交才有此字段 |
| dealNum | | Integer | 涉及到的撮合币种，挂单成交才有此字段 |

## 交易效果effects说明

effects是每个用户交易记录信息里面的交易效果，是个JSON数组，数组可以包含多项，每项内容都包含效果类型effect字段，根据effect的不同里面的内容也不同：

1. offer\_funded，被动成交；参考正式环境hash：D9A717FC8388053764908DE867E4FFD72D8D3CD9AA92A53A5FF402B3313D06B0，账号：jKCQAZwwN2sQG3Mb56GmWVqxkgpLwwAZuR，其中包含的信息有：

|  |  |  |  |
| --- | --- | --- | --- |
| 字段 | | 类型 | 说明 |
| effect | | String | offer\_funded |
| counterparty | | Object | 对家信息 |
|  | account | String | 对家账号 |
| seq | Integer | 对家单子序号 |
| hash | String | 对家交易hash |
| got | | Object | 用户获得的金额 |
|  | value | String | 金额 |
| currency | String | 货币 |
| issuer | String | 货币发行方，SWT为空字符串 |
| paid | | Object | 用户付出的金额 |
|  | value | String | 金额 |
| currency | String | 货币 |
| issuer | String | 货币发行方，SWT为空字符串 |
| type | | String | 交易类型，sold或bought |
| app | | Integer | 应用来源序号（正整数），设置了才有此字段 |
| rate | | Integer | 手续费汇率，设置了才有此字段 |
| seq | | Integer | 挂单序号，表示被成交的单子 |
| price | | String | 价格，4位小数 |
| deleted | | Boolean | 单子是否被删除了，被吃了的单子会被删除掉 |

1. offer\_partially\_funded，交易部分成交；参考正式环境交易hash：80A4C0EC32AE6E16A6025A1420DD6AB67DEE2AFED1A618A71A8AD1E37A54FB2D，账号：jKCQAZwwN2sQG3Mb56GmWVqxkgpLwwAZuR；其中包含的信息有：

|  |  |  |  |
| --- | --- | --- | --- |
| 字段 | | 类型 | 说明 |
| effect | | String | offer\_partially\_funded |
| counterparty | | Object | 对家信息 |
|  | account | String | 对家账号 |
| seq | Integer | 对家单子序号 |
| hash | String | 对家交易hash |
| remaining | | Boolean | 是否有剩余的单子 |
| gets | | Object | 对方获得的金额 |
|  | value | String | 数值 |
| currency | String | 货币 |
| issuer | String | 货币发行方，SWT为空字符串 |
| pays | | Object | 对方支付的金额 |
|  | value | String | 数值 |
| currency | String | 货币 |
| issuer | String | 货币发行方，SWT为空字符串 |
| paid | | Object | 用户支付的金额 |
|  | value | String | 数值 |
| currency | String | 货币 |
| issuer | String | 货币发行方，SWT为空字符串 |
| got | | Object | 用户获得的金额 |
|  | value | String | 数值 |
| currency | String | 货币 |
| issuer | String | 货币发行方，SWT为空字符串 |
| type | | String | 交易类型，sold或bought |
| app | | Integer | 应用来源序号（正整数），设置了才有此字段 |
| rate | | Integer | 手续费汇率，设置了才有此字段 |
| seq | | Integer | 挂单序号，表示被部分成交的单子 |
| price | | String | 挂单的价格，remaining为true才有 |

1. offer\_cancelled，交易单子被取消；参考正式环境交易hash：286C40DB44DF4340B8974E635A49C46475B4D3F3CE0A67E8CB8EC93B399A94B2，账号：jKCQAZwwN2sQG3Mb56GmWVqxkgpLwwAZuR；其中包含的信息有：

|  |  |  |  |
| --- | --- | --- | --- |
| 字段 | | 类型 | 说明 |
| effect | | String | offer\_cancelled |
| hash | | String | 被取消单子的hash |
| gets | | Object | 对方获得的金额 |
|  | value | String | 数值 |
| currency | String | 货币 |
| issuer | String | 货币发行方，SWT为空字符串 |
| pays | | Object | 对方支付的金额 |
|  | value | String | 数值 |
| currency | String | 货币 |
| issuer | String | 货币发行方，SWT为空字符串 |
| type | | String | 交易类型，sell或buy |
| app | | Integer | 应用来源序号（正整数），设置了才有此字段 |
| rate | | Integer | 手续费汇率，设置了才有此字段 |
| seq | | Integer | 被取消单子的序号 |
| price | | String | 挂单的价格 |
| deleted | | Boolean | 单子是否被删除，取消单子为true |

1. offer\_created，交易单子创建；参考正式环境交易hash:55DAB736BB47E940159ADA262B050A110A4EC9C5E917A08649E54DA587E76D9A，账号：jKCQAZwwN2sQG3Mb56GmWVqxkgpLwwAZuR，其中包含的信息有：

|  |  |  |  |
| --- | --- | --- | --- |
| 字段 | | 类型 | 说明 |
| effect | | String | offer\_created |
| gets | | Object | 对方获得的金额 |
|  | value | String | 数值 |
| currency | String | 货币 |
| issuer | String | 货币发行方，SWT为空字符串 |
| pays | | Object | 对方支付的金额 |
|  | value | String | 数值 |
| currency | String | 货币 |
| issuer | String | 货币发行方，SWT为空字符串 |
| type | | String | 交易类型，sell或buy |
| app | | Integer | 应用来源序号（正整数），设置了才有此字段 |
| rate | | Integer | 手续费汇率，设置了才有此字段 |
| seq | | Integer | 新建的单子序号 |
| price | | String | 挂单的价格 |

1. offer\_bought，主动成交；参考正式环境交易hash：F137C081D7B486DA7ACA8F790DE4A231E6B23281E4CA4A5A40CEAF1E43F1390A，账号：jKCQAZwwN2sQG3Mb56GmWVqxkgpLwwAZuR；其中包含的信息有：

|  |  |  |  |
| --- | --- | --- | --- |
| 字段 | | 类型 | 说明 |
| effect | | String | offer\_bought |
| counterparty | | Object | 对家信息 |
|  | account | String | 对家账号 |
| seq | Integer | 对家单子序号 |
| hash | String | 对家交易hash |
| paid | | Object | 用户支付的金额 |
|  | value | String | 数值 |
| currency | String | 货币 |
| issuer | String | 货币发行方，SWT为空字符串 |
| got | | Object | 用户获得的金额 |
|  | value | String | 数值 |
| currency | String | 货币 |
| issuer | String | 货币发行方，SWT为空字符串 |
| type | | String | 交易类型，sold或bough |
| price | | String | 价格 |
| app | | Integer | 应用来源序号（正整数），设置了才有此字段 |
| rate | | Integer | 手续费汇率，设置了才有此字段 |

1. setregularkey，设置代理账号；其中包含的信息有：

|  |  |  |
| --- | --- | --- |
| 字段 | 类型 | 说明 |
| effect | String | set\_regular\_key |
| type | String | 交易类型，setregularkey |
| account | String | 这是代理的源账号 |
| regularkey | String | 代理账号 |

# 底层常见错误附录

|  |  |
| --- | --- |
| 错误名称 | 说明 |
| tecCLAIM | Fee claimed. Sequence used. No action. |
| tecDIR\_FULL | Can not add entry to full directory. |
| tecFAILED\_PROCESSING | Failed to correctly process transaction. |
| tecINSUF\_RESERVE\_LINE | Insufficient reserve to add trust line. |
| tecINSUF\_RESERVE\_OFFER | Insufficient reserve to create offer. |
| tecNO\_DST | Destination does not exist. Send SWT to create it. |
| tecNO\_DST\_INSUF\_SWT | Destination does not exist. Too little SWT sent to create it. |
| tecNO\_LINE\_INSUF\_RESERVE | No such line. Too little reserve to create it. |
| tecNO\_LINE\_REDUNDANT | Can't set non-existent line to default. |
| tecPATH\_DRY | Path could not send partial amount. |
| tecPATH\_PARTIAL | Path could not send full amount. |
| tecMASTER\_DISABLED | Master key is disabled. |
| tecNO\_REGULAR\_KEY | Regular key is not set. |
| tecUNFUNDED | One of \_ADD, \_OFFER, or \_SEND. Deprecated. |
| tecUNFUNDED\_ADD | Insufficient SWT balance for WalletAdd. |
| tecUNFUNDED\_OFFER | Insufficient balance to fund created offer. |
| tecUNFUNDED\_PAYMENT | Insufficient SWT balance to send. |
| tecOWNERS | Non-zero owner count. |
| tecNO\_ISSUER | Issuer account does not exist. |
| tecNO\_AUTH | Not authorized to hold asset. |
| tecNO\_LINE | No such line. |
| tecINSUFF\_FEE | Insufficient balance to pay fee. |
| tecFROZEN | Asset is frozen. |
| tecNO\_TARGET | Target account does not exist. |
| tecNO\_PERMISSION | No permission to perform requested operation. |
| tecNO\_ENTRY | No matching entry found. |
| tecINSUFFICIENT\_RESERVE | Insufficient reserve to complete requested operation. |
| tecNEED\_MASTER\_KEY | The operation requires the use of the Master Key. |
| tecDST\_TAG\_NEEDED | A destination tag is required. |
| tecINTERNAL | An internal error has occurred during processing. |
| tefALREADY | The exact transaction was already in this ledger. |
| tefBAD\_ADD\_AUTH | Not authorized to add account. |
| tefBAD\_AUTH | Transaction's public key is not authorized. |
| tefBAD\_LEDGER | Ledger in unexpected state. |
| tefCREATED | Can't add an already created account. |
| tefEXCEPTION | Unexpected program state. |
| tefFAILURE | Failed to apply. |
| tefINTERNAL | Internal error. |
| tefMASTER\_DISABLED | Master key is disabled. |
| tefMAX\_LEDGER | Ledger sequence too high. |
| tefNO\_AUTH\_REQUIRED | Auth is not required. |
| tefPAST\_SEQ | This sequence number has already past. |
| tefWRONG\_PRIOR | This previous transaction does not match. |
| telLOCAL\_ERROR | Local failure. |
| telBAD\_DOMAIN | Domain too long. |
| telBAD\_PATH\_COUNT | Malformed: Too many paths. |
| telBAD\_PUBLIC\_KEY | Public key too long. |
| telFAILED\_PROCESSING | Failed to correctly process transaction. |
| telINSUF\_FEE\_P | Fee insufficient. |
| telNO\_DST\_PARTIAL | Partial payment to create account not allowed. |
| telBLKLIST | Tx disable for blacklist. |
| telINSUF\_FUND | Fund insufficient. |
| temMALFORMED | Malformed transaction. |
| temBAD\_AMOUNT | Can only send positive amounts. |
| temBAD\_AUTH\_MASTER | Auth for unclaimed account needs correct master key. |
| temBAD\_CURRENCY | Malformed: Bad currency. |
| temBAD\_EXPIRATION | Malformed: Bad expiration. |
| temBAD\_FEE | Invalid fee, negative or not SWT. |
| temBAD\_ISSUER | Malformed: Bad issuer. |
| temBAD\_LIMIT | Limits must be non-negative. |
| temBAD\_QUORUM | Quorums must be non-negative. |
| temBAD\_WEIGHT | Weights must be non-negative. |
| temBAD\_OFFER | Malformed: Bad offer. |
| temBAD\_PATH | Malformed: Bad path. |
| temBAD\_PATH\_LOOP | Malformed: Loop in path. |
| temBAD\_SEND\_SWT\_LIMIT | Malformed: Limit quality is not allowed for SWT to SWT. |
| temBAD\_SEND\_SWT\_MAX | Malformed: Send max is not allowed for SWT to SWT. |
| temBAD\_SEND\_SWT\_NO\_DIRECT | Malformed: No Skywell direct is not allowed for SWT to SWT. |
| temBAD\_SEND\_SWT\_PARTIAL | Malformed: Partial payment is not allowed for SWT to SWT. |
| temBAD\_SEND\_SWT\_PATHS | Malformed: Paths are not allowed for SWT to SWT. |
| temBAD\_SEQUENCE | Malformed: Sequence is not in the past. |
| temBAD\_SIGNATURE | Malformed: Bad signature. |
| temBAD\_SRC\_ACCOUNT | Malformed: Bad source account. |
| temBAD\_TRANSFER\_RATE | Malformed: Transfer rate must be >= 1.0 |
| temDST\_IS\_SRC | Destination may not be source. |
| temDST\_NEEDED | Destination not specified. |
| temINVALID | The transaction is ill-formed. |
| temINVALID\_FLAG | The transaction has an invalid flag. |
| temREDUNDANT | Sends same currency to self. |
| temREDUNDANTSIGN | Add self as additional sign. |
| temSKYWELL\_EMPTY | PathSet with no paths. |
| temUNCERTAIN | In process of determining result. Never returned. |
| temUNKNOWN | The transaction requires logic that is not implemented yet. |
| temDISABLED | The transaction requires logic that is currently disabled. |
| temMULTIINIT | contract code has multi init function |
| terRETRY | Retry transaction. |
| terFUNDS\_SPENT | Can't set password, password set funds already spent. |
| terINSUF\_FEE\_B | Account balance can't pay fee. |
| terLAST | Process last. |
| terNO\_SKYWELL | Path does not permit rippling. |
| terNO\_ACCOUNT | The source account does not exist. |
| terNO\_AUTH | Not authorized to hold IOUs. |
| terNO\_LINE | No such line. |
| terPRE\_SEQ | Missing/inapplicable prior transaction. |
| terOWNERS | Non-zero owner count. |
| tesSUCCESS | The transaction was applied. Only final in a validated ledger. |

# Solidity ERC20源码

pragma solidity ^0.5.4;

contract TokenTest {

string public name;

string public symbol;

uint8 public decimals = 18; // decimals 可以有的小数点个数，最小的代币单位。18 是建议的默认值

uint256 public totalSupply;

address payable public owner;

// 用mapping保存每个地址对应的余额

mapping (address => uint256) public balanceOf;

// 存储对账号的控制

mapping (address => mapping (address => uint256)) public allowance;

/\*\*

\* 初始化构造

\*/

constructor(uint256 initialSupply, string memory tokenName , string memory tokenSymbol) public {

totalSupply = initialSupply \* 10 \*\* uint256(decimals); // 供应的份额，份额跟最小的代币单位有关，份额 = 币数 \* 10 \*\* decimals。

balanceOf[msg.sender] = totalSupply;

name = tokenName; // 代币名称

symbol = tokenSymbol; // 代币符号

owner = msg.sender;

}

/\*\*

\* 代币交易转移的内部实现

\*/

function \_transfer(address \_from, address \_to, uint \_value) internal{

// 确保目标地址不为0x0，因为0x0地址代表销毁

require(\_to != address(0x0));

// 检查发送者余额

require(balanceOf[\_from] >= \_value);

// 确保转移为正数个

require(balanceOf[\_to] + \_value > balanceOf[\_to]);

// 以下用来检查交易，

uint previousBalances = balanceOf[\_from] + balanceOf[\_to];

// Subtract from the sender

balanceOf[\_from] -= \_value;

// Add the same to the recipient

balanceOf[\_to] += \_value;

// 用assert来检查代码逻辑。

assert(balanceOf[\_from] + balanceOf[\_to] == previousBalances);

}

/\*\*

\* 代币交易转移

\* 从自己（创建交易者）账号发送`\_value`个代币到 `\_to`账号

\* @param \_to 接收者地址

\* @param \_value 转移数额

\*/

function transfer(address \_to, uint256 \_value) public {

\_transfer(msg.sender, \_to, \_value);

}

function withdraw() payable public{

require(msg.sender == owner);

owner.transfer(address(this).balance);

}

function SWTBalance(address \_from) public view returns (uint256) {

return \_from.balance;

}

}