井通科技

jingtum-lib接口说明

V2.1.1

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

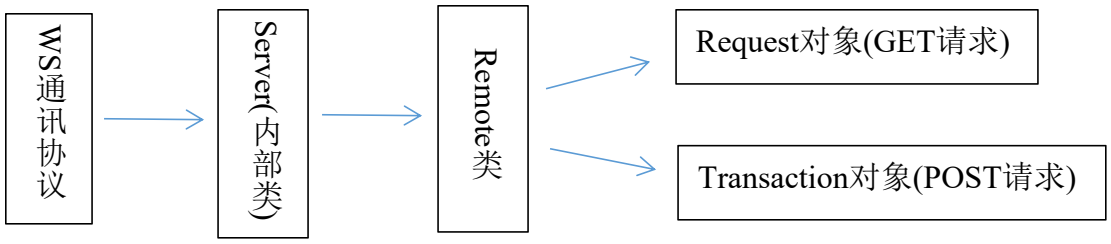
井通科技
JINGTUM TECH

1 安装

```
npm install --save jingtum-lib
```

2 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：



3 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

4 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)
* requestPathFind(options)
* createAccountStub()
* createOrderBookStub()
* buildPaymentTx(options)
* buildRelationTx(options)
* buildAccountSetTx(options)
* buildOfferCreateTx(options)
* buildOfferCancelTx(options)
* deployContractTx(options)
* callContractTx(options)

4.1 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});
```

4.2 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: 'AD8112448B3B114880D8F65F3309654A4A6FE4590839D10A5C1E8AF602338F
5D',
  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

4.3 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
    }
});
```

4.4 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

4.5 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: '945DA5D61F9AA986C784B6353C7169BC981315CDBC05922C4F34E41B8
A45B19F',

  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

4.6 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

4.7 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: '084C7823C318B8921A362E39C67A6FB15A
DA5BCCD0C7E9A3B13485B1EF2A4313'});
    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: '021EAF8D79442EF3C27B6398DB39D37F9593F2E5953CF39E57E9A10
9D4875A
7017',
  Timestamp: 572980620,
  TransactionType: 'Payment',
  TxnSignature: '30440220256C52A2C921D1D13FEC048BBFCB1F638470125244F84849
5E43C99
0590269FD02206B0D0D2CB465DB367DD3677C8BA3713DFD47BAA94D630573218DA707AA93
10A5',
```

```
  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

4.8 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: 'AE11268454C05B4A2FFFBEC53FFC8FC92118DE387B2911A00879
E6BD76C27D96',
     PreviousTxnLgrSeq: 8642907,
     RegularKey: 'jp53tPyrQLoFriTJhtm8Z9iLUXUDucnwVk',
     Sequence: 1934,
     TransferRate: 1800000000,
     index: 'E80FF91725E82A623ADC46B458D37FB270651A76A07CD6C8115F12028563
42E6' },
  ledger_hash: '79F1CB1EEF303E8928635AA68E4E0319898B7C113AC4C183CD591F274
F0FE2D4',
  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

4.9 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: '870FD292E341D25CEEC919E68FAE4C0E40F6E46045226EED3793B34F0
8030688',

  ledger_index: 8643336,

  receive_currencies:

   [ 'AAA',

     'CCA',

     'CNY',

     'USD',

     '810000000100002016002200000000020000001',
```

```
    '8100000036000020160622201606300120000002',
    '8300000001000020160021201600240020000001',
    '830000000B0000201600040000000000020000001' ],
  send_currencies:
   [ 'AAA',
    'CCA',
    'CNY',
    'USD',
    '8100000001000020160022000000000020000001',
    '8100000036000020160622201606300120000002',
    '8300000001000020160021201600240020000001',
    '830000000B0000201600040000000000020000001' ],
  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

4.10 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:'tr
ust'};
    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: 'EFC95414F74F4DFC27703E68FAB28186818A5F596D5CD105EC295B34C
7E5A70D',
  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

4.11 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: 'A88647AE9459F2E9538190D857D82437003C21769DE22D6A2F041A33F
83E2753',
  ledger_index: 8643476,
  offers:
   [ { flags: 131072,
       seq: 206,
       taker_gets: {
         currency: 'USD',
         issuer: 'jBciDE8Q3uJjf111VeiUNM775AMKHEbBLS',
         value: '0.5' },
       taker_pays: '10000000' },
     { 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
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

4.12 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: [] },
```

```
    ... 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

4.13 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: 'jBciDE8Q3uJjf111VeiUNM775AMKHEb
BLS' }};

    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: '61EF253552DDD8E9D5A1071C81591086E0132391F92311BB5B
03BD5CE35839AA',
       BookNode: '0000000000000000',
       Flags: 0,
       LedgerEntryType: 'Offer',
       OwnerNode: '0000000000000000',
       PreviousTxnID: '547266B62E7B0931F829C58B89541BCFE94ECB1EA78E712D8D
AD071D40A1EDB9',
       PreviousTxnLgrSeq: 8574138,
       Sequence: 11,
       TakerGets: [Object],
       TakerPays: '1028266745029520',
       index: 'CDD0F0DE4D63DB7F208B6EF5A84D2FE5A8CDD53DB68B876A17E9EE357E
FC877B',
       owner_funds: '988270574.4452997',
       quality: '1052631.578982826' },
     { Account: 'jKog7BTbU7wzDDDH3FPPmveszWQnZSeP5W',
       BookDirectory: '61EF253552DDD8E9D5A1071C81591086E0132391F92311BB5B
03C78C2C91CEFB',
       BookNode: '0000000000000000',
       Flags: 0,
       LedgerEntryType: 'Offer',
       OwnerNode: '0000000000000000',
       PreviousTxnID: 'BBB6A770A818427EC3B3778F94EF1577704EFCAD0C7602740E
304207E9FC4943',
       PreviousTxnLgrSeq: 1352529,
       Sequence: 10,
       TakerGets: [Object],
```

```
      TakerPays: '100000000',
      index: '9EDE0EDF25FA1AAC26B11DB38F90095AC8A84351B4B83773162A5B76BF
F98674',
      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

4.14 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',
  currency: 'TES',
  issuer: 'jBciDE8Q3uJjf111VeiUNM775AMKHEbBLS',
  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
currency	String	货币种类
issuer	String	货币发行方
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

4.15 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。

4.15.1 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

4.15.2 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

4.15.3 设置备注

方法：tx.addMemo(memo);

参数：

参数	类型	说明
memo	String	备注信息

4.15.4 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 va
lidated ledger.',
  tx_blob: '1200002200000000240000079261400000000007A1206840000000000271
0732102FE64E0C20F0058F22F3742EDC15F49F318C04F88B130742C68BAF3B1C89FD16774
46304402202EC74EFA5D3AAA663B49610B45E92438C5FD206376CB23323726180656C9EFF
9022038577D38CA50296A198A4CD2BE06BCFD8916A64E40D68F5C91C12110A92466988114
72F05993EBA9858291D364EBF6EEC3D851BD3792831485B6C98BAD6DBF7805D3C5CCC1B4F
989E0CE6749F9EA7D32E7BB996A44556A716F445A4C687A7834444366367076536976 6A6B
6A6774524553593632 63E694AFE4BB98302E357377742EE1F1',
  tx_json:
   { Account: 'jB7rxgh43ncbTX4WeMoeadiGMfmfqY2xLZ',
     Amount: '500000',
     Destination: 'jDUjqoDZLhzx4DCf6pvSivjkjgtRESY62c',
     Fee: '10000',
     Flags: 0,
     Memos: [ [Object] ],
     Sequence: 1938,
     SigningPubKey: '02FE64E0C20F0058F22F3742EDC15F49F318C04F88B130742C68
BAF3B1C89FD167',
     TransactionType: 'Payment',
     TxnSignature: '304402202EC74EFA5D3AAA663B49610B45E92438C5FD206376CB2
3323726180656C9EFF9022038577D38CA50296A198A4CD2BE06BCFD8916A64E40D68F5C91
C12110A9246698',
     hash: 'FA886D470C6B3A96FBBD57639325AEFB9104B56D912AE8AE9E385914EA63E
6B6' }
```

```
}
```

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

4.16 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)。

4.16.1 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

4.16.2 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

4.16.3 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 va
lidated ledger.',

   tx_blob: '120015220000000024000007932F223CA4FC2023000000163D4038D7EA4
C6800000000000000000000000000000434341000000000001B1CC46616D04D5A8EB8397A7C6
0B623521336DA6840000000000002710732102FE64E0C20F0058F22F3742EDC15F49F318C0
4F88B130742C68BAF3B1C89FD16774463044022064FE815BAB29F691BC5515AB409AF5866
A75FA474C0018F33B0D2CF1500170BE02202EA9CA5DAD91922A9785C8837C9C1037E0BEB1
E0818EA63FA102F558C6ED126C811472F05993EBA9858291D364EBF6EEC3D851BD3792871
485B6C98BAD6DBF7805D3C5CCC1B4F989E0CE6749',

  tx_json:

   { Account: 'jB7rxgh43ncbTX4WeMoeadiGMfmfqY2xLZ',

     Fee: '10000',

     Flags: 0,

     LimitAmount:

      { currency: 'CCA',

        issuer: 'js7M6x28mYDiZVJJtfJ84ydrv2PthY9W9u',

        value: '0.01' },

     RelationType: 1,

     Sequence: 1939,

     SigningPubKey: '02FE64E0C20F0058F22F3742EDC15F49F318C04F88B130742C68
BAF3B1C89FD167',

     Target: 'jDUjqoDZLhzx4DCf6pvSivjkjgtRESY62c',

     Timestamp: 574399740,

     TransactionType: 'RelationSet',

     TxnSignature: '3044022064FE815BAB29F691BC5515AB409AF5866A75FA474C001
8F33B0D2CF1500170BE02202EA9CA5DAD91922A9785C8837C9C1037E0BEB1E0818EA63FA1
02F558C6ED126C',

     hash: '2BC2051D97999852E515A0E5733F3C0FCC21E0270D93C5CEB43D80BDC1757
35A' }

}
```

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

4.17 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。

4.17.1 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

4.17.2 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

4.17.3 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);
        }
    });
});
```

4.18 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。

4.18.1 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

4.18.2 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

4.18.3 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 va
lidated ledger.',
   tx_blob: '1200072200080000240000007982F223CAB026440000000000F424065D403
8D7EA4C6800000000000000000000000000434E5900000000007478E561645059399B334
448F7544F2EF308ED32684000000000002710732102FE64E0C20F0058F22F3742EDC15F49
F318C04F88B130742C68BAF3B1C89FD1677446304402205E9C26D6F0C4ECC043FAC2D7ADA
9740763C89B999857C56047BD3AD3F758976A0220501A401B89560B1F4FA2E9E7727674C0
882AA4935748B3A4CBC5BA63F17CCE40811472F05993EBA9858291D364EBF6EEC3D851BD3
792',
  tx_json:
   { Account: 'jB7rxgh43ncbTX4WeMoeadiGMfmfqY2xLZ',
     Fee: '10000',
     Flags: 524288,
     Sequence: 1944,
     SigningPubKey: '02FE64E0C20F0058F22F3742EDC15F49F318C04F88B130742C68
BAF3B1C89FD167',
     TakerGets:
      { currency: 'CNY',
        issuer: 'jBciDE8Q3uJjf111VeiUNM775AMKHEbBLS',
        value: '0.01' },
     TakerPays: '1000000',
     Timestamp: 574401282,
     TransactionType: 'OfferCreate',
```

```
    TxnSignature: '304402205E9C26D6F0C4ECC043FAC2D7ADA9740763C89B999857C
56047BD3AD3F758976A0220501A401B89560B1F4FA2E9E7727674C0882AA4935748B3A4CB
C5BA63F17CCE40',
    hash: 'F6B113B1513CE6F5B7CB130028FD06E564E35CC7E83876460E9A9F0038365
424' }
}
```

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

4.19 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。

4.19.1 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

4.19.2 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

4.19.3 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: '120008220000000024000079A2F223CADE8201900000798684000000000
002710732102FE64E0C20F0058F22F3742EDC15F49F318C04F88B130742C68BAF3B1C89FD
16774463044022072 33C0103866BE592885745BDCD7EB501A49B704EAE21DBF5F2EC31062
52F12C022003DFB50DD42F0F82BA1532C977853929615690A7261986903301CF15611CEEA
0811472F05993EBA9858291D364EBF6EEC3D851BD3792',

  tx_json:

   { Account: 'jB7rxgh43ncbTX4WeMoeadiGMfmfqY2xLZ',

     Fee: '10000',

     Flags: 0,

     OfferSequence: 1944,

     Sequence: 1946,

     SigningPubKey: '02FE64E0C20F0058F22F3742EDC15F49F318C04F88B130742C68
BAF3B1C89FD167',

     Timestamp: 574402024,

     TransactionType: 'OfferCancel',

     TxnSignature: '304402207233C0103866BE592885745BDCD7EB501A49B704EAE21
DBF5F2EC3106252F12C022003DFB50DD42F0F82BA1532C977853929615690A72619869033
01CF15611CEEA0',

     hash: '44BD366DF7BDE52C52FB37B7653564BF685B84B5E51A70E23B0BDCB26D4BA
90F' }

}
```


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

4.20 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。

4.20.1 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

4.20.2 传入密钥

方法：tx.setSecret(secret);

参数：

参数	类型	说明
secret	String	源账号私钥

4.20.3 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.stri
ngToHex('result={};  function Init(t)  result=scGetAccountBalance(t)  ret
urn 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 va
lidated ledger.',
  tx_blob: '12001E22000000000240000004FA2024000000006140000000000989680684000
0000000027107321033 0E7FC9D56BB25D6893BA3F317AE5BCF33B3291BD63DB32654A313
222F7FD02074473045022100E53D151ED6EEC46124CB75C0ED0DCACC7D4458B5610925243
1DAA80D11D1099802207FE4C2987AFBBD7F1FEC6DBFBD9A32C35D2F82CEC44E5AA6EE3DE0
E58F1D8B627F94726573756C743D7B7D3B202066756E6374696F6E20496E6974287429202
0726573756C743D73634765744163636F756E7442616C616E6365287429202072657475726
6E20726573756C742020656E643B202066756E6374696F6E20666F6F28742920207265737
56C743D73634765744163636F756E7442616C616E6365287429202072657475726E207265
73756C742020656E648114B5F762798A53D543A014CAF8B297CFF8F2F937E8FAEB7012226
A486239434A41577942346A7239315652576E3936446B756B4734627764747954 68E1F1',
  tx_json:
   { Account: 'jHb9***tyTh',
     Amount: '10000000',
     Args: [ [Object] ],
     Fee: '10000',
     Flags: 0,
     Method: 0,
```

```
    Payload: '726573756C743D7B7D3B202066756E6374696F6E20496E697428742920
20726573756C743D73634765744163636F756E7442616C616E6365287429202072657475726
26E20726573756C742020656E643B202066756E6374696F6E20666F6F2874292020726573
756C743D73634765744163636F756E7442616C616E6365287429202072657475726E20726
573756C742020656E64',

    Sequence: 1274,

    SigningPubKey: '0330E7FC9D56BB25D6893BA3F317AE5BCF33B3291BD63DB32654
A313222F7FD020',

    TransactionType: 'ConfigContract',

    TxnSignature: '3045022100E53D151ED6EEC46124CB75C0ED0DCACC7D4458B5610
9252431DAA80D11D1099802207FE4C2987AFBBD7F1FEC6DBFBD9A32C35D2F82CEC44E5AA6
EE3DE0E58F1D8B62',

    hash: 'D7E40A7164C11BFA81C05117010631EA5BCBD8A1A0B3B2FF343D9FB3F3575
936'  }

}
```


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

4.21 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。

4.21.1 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
foo	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

4.21.2 传入密钥

方法：tx.setSecret(secret);

参数：

参数	类型	说明

secret	String	源账号私钥

4.21.3 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',foo: '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 va
lidated ledger.',
  tx_blob: '12001E22000000002400000 4FC2024000000016840000000000027107321 0
330E7FC9D56BB25D6893BA3F317AE5BCF33B3291BD63DB32654A313222F7FD02074473045
022100835BEDFD6794F3B95A31971EDBF1865F9190D1A66FFC9515883B15107B368894022
052F44E0E16B6A3BC3C8CF2A3881E35687060EB73B6B0503611CEF58781A512EA70110366
6F6F8114B5F762798A53D543A014CAF8B297CFF8F2F937E88314EC4F411C3B37C3E451D46
2C27489B0549650572CFAEB7012226A486239434A41577942346A7239315652576E393644
6B756B473462776474795468E1F1',
  tx_json:
   { Account: 'jHb9***tyTh',
     Args: [ [Object] ],
     ContractMethod: '666F6F',
     Destination: 'j4YVQxCxaRRQ6gCVUvi9MoiTfWyPRnHwej',
     Fee: '10000',
     Flags: 0,
     Method: 1,
     Sequence: 1276,
     SigningPubKey: '0330E7FC9D56BB25D6893BA3F317AE5BCF33B3291BD63DB32654
A313222F7FD020',
     TransactionType: 'ConfigContract',
     TxnSignature: '3045022100835BEDFD6794F3B95A31971EDBF1865F9190D1A66FF
C9515883B15107B368894022052F44E0E16B6A3BC3C8CF2A3881E35687060EB73B6B05036
11CEF58781A512EA',
     hash: '8DE51A01F6FA55F5FDBB196A39A1F4220772E19835C81CE2E8A44B487BCD9
61C' }
```

```
}
```

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

4.22 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。

4.22.1 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
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

4.22.2 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

4.22.3 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,
        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 va
lidated ledger.',
  tx_blob: '1200CD22000000002400000001202500000001390000000000000000013A000
00000000003E861D38AA87BEE53800000000000000000000000054455354000000000074
78E561645059399B334448F7544F2EF308ED32684000000000002710732103 30E7FC9D56B
B25D6893BA3F317AE5BCF33B3291BD63DB32654A313222F7FD02074473045022100FE5130
87425A863D1FB0004B29C6656E1B4C237F57F193FC5707317257926787022070AA7125CB5
383F5B53000903BDAF9926CBE74F732DF7E09BE3EC581FEF5E7C08114B5F762798A53D543
A014CAF8B297CFF8F2F937E8',
  tx_json:
   { Account: 'jHb9CJAWyB4jr91VRWn96DkukG4bwdtyTh',
     Amount:
      { currency: 'TEST',
        issuer: 'jBciDE8Q3uJjf111VeiUNM775AMKHEbBLS',
        value: '3' },
     AppType: 1,
     Fee: '10000',
     Flags: 0,
     OfferFeeRateDen: '00000000000003E8',
     OfferFeeRateNum: '0000000000000001',
     Sequence: 1,
     SigningPubKey: '0330E7FC9D56BB25D6893BA3F317AE5BCF33B3291BD63DB32654
A313222F7FD020',
     TransactionType: 'Brokerage',
     TxnSignature: '3045022100FE513087425A863D1FB0004B29C6656E1B4C237F57F
193FC5707317257926787022070AA7125CB5383F5B53000903BDAF9926CBE74F732DF7E09
BE3EC581FEF5E7C0',
```

```
    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
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

4.23 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。

4.23.1 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

4.23.2 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

4.23.3 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 req = remote.initContract({
        account: v.address,
        amount: 10,
        payload: '606060405260126002600060101000a81548160ff021916908360ff1
602179055503415610002b57600080fd5b6040516109273803806109278339810160405280
805190602001909190805182019190602001805182019190505060026000905490610100
a900460ff1660ff16600a0a830260038190555060035460046000337 3fffffffffffffffff
ffffffffffffffffffffffff1673ffffffffffffffffffffffffffffffffffffffff16815
260200190815260200160002081905550816000908051906020019061 00d39291906100f3
565b5080600190805190602001906100ea9291906100f3565b5050505061019856 5b82805
460018160011615610100020316600290049060005260206000209060 1f01602090048101
9282601f10610134578051 60ff191683800117855561016256 5b8280016001018555821561
01625791820 15b8281111561016157825182559160200191906001019061 0146565b5b50
905061016f9190610173565b5090565b6101959190 5b808211156101915760008160009055
```

```
55060010161017956 5b5090565b90565b610780806101a76000396000f300606060405260
0436106100835760003 57c0100000000000000000000000000000000000000000000000000
0000000900463ffffffff16806306fdde0314610093578063 18160ddd1461012157806331
3ce5671461014a57806370a08231146101795780639 5d89b41146101c6578063a9059cbb1
4610254578063dd62ed3e1461029657 5b341561008e57600080fd5b600080fd5b34156100
9e57600080fd5b6100a6610302565b6040518080602001828103825283818151815260200
191508051906020019080838360005b838110156100e6578082015181840152602081019
506100cb565b50505050905090810190601f168015610113578082038051600183602003 6
101000a031916815260200191505b509250505060405180910390f35b341561012c576000
80fd5b6101346103a0565b6040518082815260200191505060405180910390f35b3415610
15557600080fd5b61015d6103a6565b604051808260ff1660ff16815260200191505060405
5180910390f35b34156101845760080fd5b6101b0600480803573ffffffffffffffffffff
ffffffffffffffffffff1690602001909190505061 03b9565b6040518082815260200191
505060405180910390f35b34156101d157600080fd5b6101d96103d1565b6040518080602
0018281038252838181518152602001915080519060200190808383600 05b838110156102
1957808201518184015260208101905061 01fe565b505050509050908101906 01f1680156
1024657808203805160018360200361 01000a031916815260200191505b50925050506040
5180910390f35b341561025f57600080fd5b610294600480803573ffffffffffffffffffff
ffffffffffffffffffff1690602001909190803590602001909190505061046f565b005b
34156102a157600080fd5b6102ec600480803573ffffffffffffffffffffffffffffffffff
ffffff1690602001909190803573ffffffffffffffffffffffffffffffffffffffffffff1690
6020019091905050 61047e565b6040518082815260200191505060405180910390f35b600
0805460018160011615610100020316600290048060 1f0160208091040260200160405190
8101604052809291908181526020018280546001816001161561010002031660029004801
56103985780601f1061036d576101008083540402835291602001916103985 65b82019190
6000526020600020905b81548152906001019060200180831161037b57829003601f16820
1915b505050505081565b60035481565b6002600090549061010000a900460ff1681565b60
0460205280600052604060002060009150905054815 65b6001805460018160011615610100
0020316600290048060 1f0160208091040260200160405190810160405280929190818152
6020018280546001816001161561010002031660029004801561046757 80601f1061043c5
7610100808354040283529160200191610467565b8201919060005260206000209 05b8154
8152906001019060200180831161044a57829003601f168201915b505050505081565b610
47a3383836104a3565b5050565b600560205281600052604060002060205280600052604 0
6000206000915091505054815 65b6000808373ffffffffffffffffffffffffffffffffff
fffff16141515156104ca57600080fd5b81600460008673ffffffffffffffffffffffffffff
ffffffffffff1673ffffffffffffffffffffffffffffffffffffffffffffffff168152602001908
1526020016000205410151515610518576000 80fd5b600460008473ffffffffffffffffff
ffffffffffffffffffff1673ffffffffffffffffffffffffffffffffffffffffffffffff1681526
0200190815260200160002054826004600086 73ffffffffffffffffffffffffffffffffff
ffffff1673ffffffffffffffffffffffffffffffffffffffffffff168152602001908152602 00
160002054011115156105a657600080fd5b600460008473ffffffffffffffffffffffffffff
ffffffffffff1673ffffffffffffffffffffffffffffffffffffffffffffffff168152602001908
```

```
1526020016000205460046000867 3fffffffffffffffffffffffffffffffffffffff1673
ffffffffffffffffffffffffffffffffffffffff16815260200190815260200160002054 0
190508160046000867 3fffffffffffffffffffffffffffffffffffffff1673ffffffffff
ffffffffffffffffffffffffffff16815260200190815260200160002060008282540 39
250508190555081600460008573ffffffffffffffffffffffffffffffffffffff1673ff
ffffffffffffffffffffffffffffffffffff16815260200190815260200160002060008
282540192505081905550806004600085 73ffffffffffffffffffffffffffffffffffff
ff1673ffffffffffffffffffffffffffffffffffff16815260200190815260200160 0
020546004600087 73ffffffffffffffffffffffffffffffffffff1673ffffffffffff
ffffffffffffffffffffffffff16815260200190815260200160002054 0114151561074
e57fe5b505050505600a165627a7a72305820b2665df0d8d8522803a19ac6bc98ff010121
e11c16d0342eaced01d94100ce180029',
       abi: abi,
       params:[2000, 'TestCurrency', 'TEST1']
    });
    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：

```
{
  ContractState: 'jPZ1....9Kkh',
  engine_result: 'tesSUCCESS',
  engine_result_code: 0,
```

```
  engine_result_message: 'The transaction was applied. Only final in a va
lidated ledger.',

  tx_blob: '12001F220000000024000019D2F2410A4D52024000000006140000000009
896806840000000000027107321033 0E7FC9D56BB25D6893BA3F317AE5BCF33B3291BD63D
B32654A313222F7FD0207446304402206C941CF3355094C5E85C2410FBB131288802094B7
90C399639A543A9DD64DDB1022059EEA991A946A28EA9695AEB2F5F724831C371BB0F9C72
D50DD6855BCDF5C07D7FD44D3630363036303430353236303132363030323630303036313
031303030613831353438313630666630323139313639303833363066663136303231373 9
303535353033343135363130303262353736303030383066643562363034303531363130 3
932373338303338303631303932373833333938313031363034303532383038303531393 0
363032303031393039313930383035313832303139313930363032303031383035313832 3
031393139303530353036303032363030303930353439303631303130303061393030343 6
306666313636306666313636303061306138333032363030333831393035353530363030 3
335343630303436303030303333373366666666666666666666666666666666666666666 6
666666666666666666666666666666666666666313637336666666666666666666666666 6
666666666666666666666666666666666666666666666666666666313638313532363032303 0
313930383135323630323030313630303032303831393035353530383136303030393038 3
035313930363032303031393036313030643339323931393036313030663335363562353 0
383036303031393038303531393036303230303139303631303065613932393139303631 3
030663335363562353035303530353036313031393835363562383238303534363030313 8
313630303131363135363130313030303230333136363030323930303439303630303035 3
236303230363030303230393036303166303136303230393030343831303139323832363 0
316631303631303133343537383035313630666631393136383338303031313738353535 3
631303136323536356238323830303136303031303138353535383231353631303136323 5
373931383230313562383238313131313536313031363135373832353138323535393136 3
032303031393139303630303130313930363130313436353635623562353039303530363 1
303136663931393036313031373335363562353039303536356236313031393539313930 3
562383038323131313536313031393135373630303038313630303039303535353036303 0
313031363130313739353635623530393035363562393035363562363130373830383036 3
130316137363030303339363030306633303036303630363034303532363030343336313 0
363130303833353736303030333537633031303030303030303030303030303030303030 3
030303030303030303030303030303030303030303030303030303030303030303030303 0
303930303436336666666666666666313638303633303666646465303331343631303039 3
335373830363331383136306464643134363130313231353738303633333133636535363 7
313436313031346135373830363337306130383233331313436313031373935373830363 3
935643839623431313436313031633635373830363361393035396362623134363130323 5
343537383036336464363265643365313436313032393635373562333431353631303038 6
535373630303038306664356236303030383066643562333431353631303039653537363 0
303038306664356236313030613636313033303235363562363034303531383038303630 3
230303138323831303338323532383338313831353138313532363032303031393135303 8
303531393036303230303139303830383338333630303035623833383131303135363130 3
065363537383038323031353138313834303135323630323038313031393035303631303 0
```

```
6362353635623530353035303530393035303930383130313930363031663136383031353
6313031313335373830383230333830353136303031383336303230303336313031303030
6130333139313638313532363032303031393135303562353039323530353035303630343
0353138303931303339306633356233343135363130313263353736303030383066643562
3631303133343631303361303536356236303430353138303832383135323630323030313
9313530353036303430353138303931303339306633356233343135363130313535353736
3030303830666435623631303135643631303361363536356236303430353138303832363
0666631363630666631363831353236303230303139313530353036303430353138303931
3033393066333562333431353631303138343537363030303830666435623631303162303
6303034383038303335373366666666666666666666666666666666666666666666666666
6666666666666666666666666666663136393036303230303139303931393035303530363
1303362393536356236303430353138303832383135323630323030313931353035303630
3430353138303931303339306633356233343135363130316431353736303030383066643
5623631303164393631303364313536356236303430353138303830363032303031383238
3130333832353238333831383135313831353236303230303139313530383035313930363
0323030313930383038333833363030303562383338313130313536313032313935373830
3832303135313831383430313532363032303831303139303530363130316665353635623
5303530353035303930353039303831303139303630316631363830313536313032343635
3738303832303338303531363030313833363032303033363130313030306130333139313
6383135323630323030313931353035623530393235303530353036303430353138303931
3033393066333562333431353631303235663537363030303830666435623631303239343
6303034383038303335373366666666666666666666666666666666666666666666666666
6666666666666666666666666666663136393036303230303139303931393038303335393
0363032303031393039313930353035303631303436663536356230303562333431353631
3032613135373630303038306664356236313032656336303034383038303335373366666
6666666666666666666666666666666666666666666666666666666666666666666666666
6631363930363032303031393039313930383033353733666666666666666666666666666
6666666666666666666666666666666666666666666666666666666631363930363032303031
3930393139303530353036313034376535363562363034303531383038323831353236303
2303031393135303530363034303531383039313033393066333562363030303830353436
3030313831363030313136313536313031303030323033313636303032393030343830363
0316630313630323038303931303430323630323030313630343035313930383130313630
3430353238303932393139303831383135323630323030313832383035343630303138313
6303031313631353631303130303032303331363630303239303034383031353631303339
3835373830363031663130363130333664353736313031303038303833353430343032383
3353239313630323030313931363130333938353635623832303139313930363030303532
3630323036303030323039303562383135343831353239303630303130313930363032303
0313830383331313631303337623537383239303033363031663136383230313931356235
3035303530353035303831353635623630303335343831353635623630303236303030393
0353439303631303130303061393030343630666631363831353635623630303436303230
3532383036303030353236303430363030303230363030303931353039303530353438313
5363562363030313830353436303031383136303031313631353631303130303032303331
```

```
3636303032393030343830363031663031363032303830393130343032363032303031363
0343035313930383130313630343035323830393239313930383138313532363032303031
3832383035343630303138313630303131363135363130313030303230333136363030323
9303034383031353631303436373537383036303166313036313034336335373631303130
3038303833353430343032383335323931363032303031393136313034363735363562383
2303139313930363030303532363032303630303032303930356238313534383135323930
3630303130313930363032303031383038333131363130343461353738323930303336303
1663136383230313931356235303530353035303530383135363562363130343761333338
3338333631303461333536356235303530353635623630303536303230353238313630303
0353236303430363030303230363032303532383036303030353236303430363030303230
3630303039313530393135303530353438313536356236303030383038333733666666666
6666666666666666666666666666666666666666666666666666666666666666666666666631
3631343135313531353631303463613537363030303830666435623831363030343630303
0383637336666666666666666666666666666666666666666666666666666666666666666666
6666666666666666631363733666666666666666666666666666666666666666666666666666
6666666666666666666666666666666631363831353236303230303139303831353236303032
3030313630303032303534313031353135313536313035313835373630303038306664356
2363030343630303038343733666666666666666666666666666666666666666666666666666
6666666666666666666666666666666631363733666666666666666666666666666666666666
6666666666666666666666666666666666666666666666631363831353236303230303139
3038313532363032303031363030303230353438323630303436303030303836373366666666
6666666666666666666666666666666666666666666666666666666666666666666666666666
3136373366666666666666666666666666666666666666666666666666666666666666666666
6666666666666631363831353236303230303139303831353236303230303136303030320
3035343031313131353135363130356136353736303030383066643562363030343630303
0383437336666666666666666666666666666666666666666666666666666666666666666666
6666666666666666631363733666666666666666666666666666666666666666666666666666
6666666666666666666666666666666631363831353236303230303139303831353236303230
3030313630303032303534363030343630303038363733666666666666666666666666666666
6666666666666666666666666666666666666666666666666666631363733666666666666666
6666666666666666666666666666666666666666666666666666666666666666666631363733
9303831353236303230303136303030323036303030383238323534303339323530353038
3139303535353038313630303436303030383537336666666666666666666666666666666666
6666666666666666666666666666666666666666666666666666631363733666666666666666
6666666666666666666666666666666666666666666666666666666666666666666313638313
5323630323030313930383135323630323030313630303032303630303038323832353430
3139323530353038313930353535303830363030343630303038353733666666666666666666
6666666666666666666666666666666666666666666666666666666666666666666631363733
```

```
6666666666666666666666666666666666666666666666666666666666666666666666666666
6666666313638313532363032303031393038313532363032303031363030303230353436
3030343630303038373733666666666666666666666666666666666666666666666666666666
6666666666666666666666666666631363733666666666666666666666666666666666666666
6666666666666666666666666666666666666666666631363831353236303230303139303
8313532363032303031363030303230353430313134313531353631303734653537666535
6235303530353035303536303061313635363237613761373233303538323062323636356
4663064386438353232383033613139616336626339386666303130313231653131633136
6430333432656163656430316439343130306365313830303239303030303030303030303
0303030303030303030303030303030303030303030303030303030303030303030303030
3030303030303030303030303030303030303030303030303030303030303030303030303
0303030303030303030303030303030303030303030303030303030303030303030303030
3030303030303630303030303030303030303030303030303030303030303030303030303
0303030303030303030303030303030303030303030303030303030303030303030303030
3030303030303030303030303030303030303030303030303030303030303030303030303
0303030303030303030303030303030303030303030303030303030303030303030303030
3030303030303030303030303030303030303030303030303030303030303030303030303
0303030303030303535343435353335343331303030303030303030303030303030303030
3030303030303030303030303030303030303030303030303030303030303030303030308
114B5F762798A53D543A014CAF8B297CFF8F2F937E8',
  tx_json:
   { Account: 'jHb9....tyTh',
     Amount: '10000000',
     Fee: '10000',
     Flags: 0,
     Method: 0,
     Payload: '3630363036303430353236303132363030323630303036313031303030
6138313534383136306666303231393136393038333630666631363032313739303535353
0333431353631303032623537363030303830666435623630343035313631303932373338
3033383036313039323738333339383130313630343035323830383035313930363032303
0313930393139303830353138323031393139303630323030313830353138323031393139
3035303530363030323630303039303534393036313031303030613930303436306666313
6363066663136363030613061383330323630303338313930353535303630303335343630
3034363030303333373366666666666666666666666666666666666666666666666666666
6666666666666666666666666666631363733666666666666666666666666666666666666
6666666666666666666666666666666666666666666631363831353236303230313930383
```

```
1353236303230303136303030323038313930353535303831363030303930383035313930
3630323030313930363130306433393239313930363130306633353635623530383036303
0313930383035313930363032303031393036313030656139323931393036313030663335
3635623530353035303530363130313938353635623832383035343630303138313630303
1313631353631303130303032303333136363030323930303439303630303035323630323
3630303032303930363031663031363032303930303438313031393238323630316631303
6313031333435373830353136306666313931363833383030313137383535353631303136
3235363562383238303031363030313031383535353832313536313031363235373931383
2303135623832383131313135363130313631353738323531383235353931363032303031
3931393036303031303139303631303134363536356235623530393035303631303136663
9313930363130313733335363562353039303536356236313031393539313930356238303
3231313135363130313931353736303030383136303030393035353530363030313031363
1303137393536356235303930353635623930353636562363130373830383036313031613
3630303033393630303066333030363036303630343035323630303433363130363130303
8333537363030303335376330313030303030303030303030303030303030303030303030
3030303030303030303030303030303030303030303030303030303030303030303930303
4363366666666666666666631363830363333036666464653033313436313030393335373830
3633313831363064646431343631303132313537383036333331336365353637313436313
0313461353738303633373061303832333131343631303137393537383036333935643839
6234313134363130316336353738303633613930353963626231343631303235343537383
0363364643632656433653134363130323936353735623334313536313030386535373630
3030383066643562363030303830666435623334313536313030396535373630303038306
6643562363130306136363130333032353635623630343035313830383036303230303138
3238313033383235323833383138313531383135323630323030313931353038303531393
0363032303031393038303833383336303030356238333831313031353631303065363537
3830383230313531383138343031353236303230383130313930353036313030636235363
5623530353035303530393035303930383130313930363031663136383031353631303131
3335373830383230333830353136303031383336303230303336313031303030613033313
9313638313532363032303031393135303562353039323530353035303630343035313830
3931303333930663335623334313536313031326335373630303038306664356236313031
3343631303361303536356236303430353138303832383135323630323030313931353035
3036303430353138303931303333930663335623334313536313031353535373630303038
0666435623631303135643631303361363535635623630343035313830383236306666313
3630666631363831353236303230303139313530353036303430353138303931303333930
6333562333431353631303138343537363030303830666435623631303162303630303438
3038303333537336666666666666666666666666666666666666666666666666666666666
6666666666666666666663136393036303230303139303931393035303530363130336239
3536356236303430353138303832383135323630323030313931353035303630343035313
8303931303333930663335623334313536313031643135373630303038306664356236313
0316439363130336431353635623630343035313830383036303230303138323831303333
8323532383338313831353138313532363032303031393135303830353139303630323030
3139303830383338333630303035623833383131303135363130323139353738303832303
13
```

5313831383430313532363032303831303139303530363130316665353635623530353035
3035303930353039303831303139303630316631363830313536313032343635373830383
2303338303531363030313833363032303033363130313030306130333139313638313532
3630323030313931353035623530393235303530353036303430353138303931303339306
6333562333431353631303235663537363030303830666435623631303239343630303438
3038303335373366666666666666666666666666666666666666666666666666666666666
6666666666666666666663136393036303230303139303931393038303335393036303230
3031393039313930353035303631303436663536356230303562333431353631303261313
5373630303038306664356236313032656336303034383038303335373366666666666666
6666666666666666666666666666666666666666666666666666666666666666666663136393
0363032303031393039313930383033353733666666666666666666666666666666666666
6666666666666666666666666666666666666666666663136393036303230303139303931 3
9303530353036313034376535363562363034303531383038323831353236303230303139
3135303530363034303531383039313033393066333562363030303830353436303031383
1363030313136313536313031303030323033313636303032393030343830363031663031
3630323038303931303430323630323030313630343035313930383130313630343035323
8303932393139303831383135323630323030313832383035343630303138313630303131
3631353631303130303032303331363630303239303034383031353631303339383537383
0363031663130363130333664353736313031303038303833353430343032383335323931
3630323030313931363130333938353635623832303139313930363030303532363032303
6303030323039303562383135343831353239303630303130313930363032303031383038
3331313631303337623537383239303033363031663136383230313931356235303530353
0353035303831353635623630303335343831353635623630303236303030393035343930
3631303130303061393030343630666631363831353635623630303436303230353238303
6303030353236303430363030303230363030303931353039303530353438313536356236
3030313830353436303031383136303031313631353631303130303032303331363630303
2393030343830363031663031363032303830393130343032363032303031363034303531
3930383130313630343035323830393239313930383138313532363032303031383238303
5343630303138313630303131363135363130313030303230333136363030323930303438
3031353631303436373537383036303166313036313034336335373631303130303830383
3353430343032383335323931363032303031393136313034363735363562383230313931
3930363030303532363032303630303032303930356238313534383135323930363030313
0313930363032303031383038333131363130343461353738323930303336303166313638
3230313931356235303530353035303530383135363562363130343761333338333833363
1303461333536356235303530353635623630303536303230353238313630303035323630
3430363030303230363032303532383036303030353236303430363030303230363030303
9313530393135303530353438313536356236303030383038333733666666666666666666
6666666666666666666666666666666666666666666666666666666666666666666663136313431 3
5313531353631303436313537363030303830666435623831363030343630303038363733
6666666666666666666666666666666666666666666666666666666666666666666666666
6666666313637336666666666666666666666666666666666666666666666666666666666
66666666666666666666663136383135323630323030313930383135323630323030313636 3

0303032303534313031353135313536313035313835373630303038306664356236303034
3630303038343733666666666666666666666666666666666666666666666666666666666666
6666666666666666666666631363733666666666666666666666666666666666666666666666
6666666666666666666666666666666666666631363831353236303230303139303831353
2363032303031363030303230353438323630303436303030383637336666666666666666
6666666666666666666666666666666666666666666666666666666666666666666313637336
6666666666666666666666666666666666666666666666666666666666666666666666666666
6666663136383135323630323030313930383135323630323030313630303032303534303
1313131353135363130356136353736303030383066643562363030343630303038343733
6666666666666666666666666666666666666666666666666666666666666666666666666666
6666663136373366666666666666666666666666666666666666666666666666666666666666
6666666666666666666666631363831353236303230303139303831353236303230303136 3
0303032303534363030343630303038363733666666666666666666666666666666666666
6666666666666666666666666666666666666666666631363733666666666666666666666
6666666666666666666666666666666666666666666666666666666631363831353236
3032303031393038313532363032303031363030303230353430313930353038313630303
4363030303836373366666666666666666666666666666666666666666666666666666666666
6666666666666666666666631363733666666666666666666666666666666666666666666666
6666666666666666666666666666666666666631363831353236303230303139303831353
3236303230303136303030323036303030383238323534303339323530353038313930353
5353038313630303436303030383537336666666666666666666666666666666666666666
6666666666666666666666666666666666666631363733666666666666666666666666666
6666666666666666666666666666666666666666666666666666666631363831353236303032
3030313930383135323630323030313630303032303630303038323832353430313932353
0353038313930353535303830363030343630303038353733666666666666666666666666
6666666666666666666666666666666666666666666666666666666631363733666666666
6666666666666666666666666666666666666666666666666666666666666666666666666631
3638313532363032303031393038313532363032303031363030303230353436303034363
0303038373733666666666666666666666666666666666666666666666666666666666666666
6666666666666666666631363733666666666666666666666666666666666666666666666666
6666666666666666666666666666666666666631363831353236303230303139303831353236
3032303031363030303230353430313134313531353631303734653537666535623530353
0353035303536303061313635363237613761373233303538323062323636356466306438
6438353232383033613139616336626339386666303130313231653131633136643033343
2656163656430316439343130306365313830303239303030303030303030303030303030
3030303030303030303030303030303030303030303030303030303030303030303030303
0303030303030303030376430303030303030303030303030303030303030303030303030
3030303030303030303030303030303030303030303030303030303030303030303030303
0303630303030303030303030303030303030303030303030303030303030303030303030
3030303030303030303030303030303030303030303030303030303030303061303030303
0303030303030303030303030303030303030303030303030303030303030303030303030
3030303030303030303030303030303030303030306335343635373337343433373537323

```
732363536653633373930303030303030303030303030303030303030303030303030303030303030
303030303030303030303030303030303030303030303030303030303030303030303030303030303
030303030303030303030303030303030303030303030303030303030303030303030303030303030
303030353534343535333534333130303030303030303030303030303030303030303030303030303
030303030303030303030303030303030303030303030303030303030303030303030303030303030',

    Sequence: 413,

    SigningPubKey: '0330E7FC9D56BB25D6893BA3F317AE5BCF33B3291BD63DB32654
A313222F7FD020',

    TransactionType: 'AlethContract',

    TxnSignature: '304402206C941CF3355094C5E85C2410FBB131288802094B790C3
99639A543A9DD64DDB1022059EEA991A946A28EA9695AEB2F5F724831C371BB0F9C72D50D
D6855BCDF5C07D',

    hash: '7BAAF8543E5E5761500CB21F5BCBCB9CA7A786704E82153481A7944E8F6E9
7F9'

}
```


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

4.24 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。

4.24.1 创建合约部署对象

方法：`remote.invokeContract({});`

参数：

参数	类型	说明
account	String	合约发布者
destination	String	合约账号
abi	Array	合约abi
func	String	合约函数名及参数

返回：Transaction对象

4.24.2 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

4.24.3 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 req = remote.invokeContract({
    account: v.address,
    destination: 'jPZ1....9Kkh',
```

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

```
{
  ContractState: '',
  engine_result: 'tesSUCCESS',
  engine_result_code: 0,
  engine_result_message: 'The transaction was applied. Only final in a va
lidated ledger.',
  tx_blob: '12001F220000000240000019E2F2410A962202400000001684000000000000
0271073210330E7FC9D56BB25D6893BA3F317AE5BCF33B3291BD63DB32654A313222F7FD0
20744730450221 00C44640B21BF99C1764042A87FCAE6877BF1188F9F8DD887969EEF7353
33E584002204A2158F843021522FF7F93940BA4C1A933E21F03B6B9E5A3F3A54C217428CD
DC8114B5F762798A53D543A014CAF8B297CFF8F2F937E88314F78A69DCD5308DB30C41724
F04EF86CA3AC74AB3FAEB701288613930353963626230303030303030303030303030303030
3030303030303030303066373861363964636435333038646233306334313732346630346566
6383663613361633734616233303030303030303030303030303030303030303030303030303030
3030303030303030303030303030303030303030303030303030303030303030303030303030303
0303066041000E1F1',
  tx_json:
   { Account: 'jHb9CJAWyB4jr91VRWn96DkukG4bwdtyTh',
```

```
    Args: [ { Arg:

    { ContractParamsType: 0,

      Parameter: '6139303539636262303030303030303030303030303030303030303030
303030303066373861363964636435333038646233306334313732346630346566383636361336163734616233303030303030303030303030303030303030303030303030303030303030303030303030303030303030303030303030303030303030303030303030303030303030303030303030303030303030303066'
 } } ],

    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

4.25 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);
    });
});
```

5 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)

5.1 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: 'jB7rxgh43ncbTX4WeMoead
iGMfmfqY2xLZ'});
    req.selectLedger("8573498");
    req.submit(function(err, result) {
        if(err) {console.log('err:',err);}
        else if(result){
            console.log('res:', result);
        }
    });
});
```

5.2 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: 'jB7rxgh43ncbTX4WeMoead
iGMfmfqY2xLZ'});
```

```
    req.submit(function(err, result) {
       if(err) {console.log('err:',err);}
       else if(result){
           console.log('res:', result);
       }
   });
});
```

6 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)

6.1 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);
```

6.2 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);
```

6.3 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');;
```

6.4 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.');
```

6.5 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);
        }
    });
```

```
});
```

7 工具类

Utils类是工具类，提供提供以下方法：

* hexToString()

* stringToHex()

* isValidAmount()

* isValidAmount0()

* parseAmount()

* isValidCurrency()

* isValidHash()

* isValidAddress()

* isValidSecret()

* affectedAccounts()

* affectedBooks()

* processTx()

7.1 16进制转字符串

方法：hexToString()

参数：无

例子：

8 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.

9 Solidity ERC20源码

```
pragma solidity ^0.4.19;

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;

    // 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
     */
    function TokenTest(uint256 initialSupply, string tokenName, string to
kenSymbol) 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
    }

        /**
     * 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 != 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() public {
        revert();
    }
}
```

10 **Solidity ERC721源码**

```
pragma solidity ^0.4.19;
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;


    // 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
     */
    function TokenTest(uint256 initialSupply, string tokenName, string 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
    }


    /**
     * 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 != 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);
    }

    /**
```

```
     * 转账
     * 从源账号`_from`发送`_value`个代币到 `_to`账号
     *
     * @param _from 发送者地址
     * @param _to 接收者地址
     * @param _value 转移数额
     */
    function transferFrom(address _from, address _to, uint256 _value) pub
lic payable returns (bool success) {
        allowance[_from][msg.sender] = _value;
        _transfer(_from, _to, _value);
        return true;
    }


    /**
     * 转基础货币SWT
     * 从合约账号发送`_value`个代币到 `_to`账号
     *
     * @param _to 接收者地址
     * @param _value 转移数额
     */
    function transferSWT(address _to, uint256 _value) public payable retu
rns (bool ){
         _to.transfer(_value);
         return true;
    }
```

```
    /**
     * 获取账号础货币(SWT)余额
     *
     * @param _from 要获取余额的账号地址
     */
    function SWTBalance(address _from) public constant returns (uint256)
 {
        return _from.balance;
}


    function() public {
        revert();
    }
}
```