# JCC\_RPC

**[一、安装 3](#_Toc244399463)**

**[二、使用说明 3](#_Toc1633041577)**

**[1 配置接口 3](#_Toc1688775979)**

[1.1 使用示例 3](#_Toc2114000301)

[1.2 获取配置数据 3](#_Toc2033602939)

**[2 交易接口 3](#_Toc1562353768)**

[2.1 使用示例 3](#_Toc1197226907)

[2.2 获取钱包余额 4](#_Toc2018337206)

[2.3 获取交易历史记录 5](#_Toc541733230)

[2.4 获取转账历史记录 6](#_Toc1727216977)

[2.5 获取当前挂单 8](#_Toc1799275940)

[2.6 创建挂单 9](#_Toc1713490173)

[2.7 取消挂单 9](#_Toc873631341)

[2.8 获取序列号 10](#_Toc776253648)

[2.9 转账 10](#_Toc531906411)

**[3 数据接口 11](#_Toc1924110863)**

[3.1 使用示例 11](#_Toc1722517915)

[3.2 获取指定币种24小时的行情数据 11](#_Toc131552198)

[3.3 获取所有币种24小时的行情数据 12](#_Toc1237119023)

[3.4 获取市场深度 13](#_Toc322749307)

[3.5 获取K线数据 15](#_Toc2051394074)

[3.6 获取分时数据 16](#_Toc2077732780)

[3.7 获取币种间汇率 16](#_Toc223249593)

## 一、安装

npm install jcc\_rpc

## 二、使用说明

### 1 配置接口

#### 1.1 使用示例

const JcConfig = require("jcc\_rpc").JcConfig

let hosts = ["jccdex.cn", "weidex.vip"]

let port = 443

let https = true

let instance = new JcConfig(hosts, port, https)

#### 1.2 获取配置数据

let res = await instance.getConfig()

返回参数: 所有服务器配置信息，参数说明如下:

exHosts：交易服务器列表

infoHosts：信息服务器列表

cfgHosts： 配置服务器列表

说明：

以下所有API请求对应的host都是基于exHosts或者infoHosts

交易接口请调用exHosts列表中服务器

信息接口请调用infoHosts列表中服务器

配置接口请调用cfgHosts列表服务器

### 2 交易接口

#### 2.1 使用示例

const JcExchange = require("jcc\_rpc").JcExchange

let hosts = ["ejia348ffbda04.jccdex.cn"]

let port = 443

let https = true

let instance = new JcExchange(hosts, port, https)

#### 2.2 获取钱包余额

let res = await instance.getBalances(address)

请求参数：

address:钱包地址

结果如下：

{

result: true,

code: "0",

data: [

{

value: "137285.0869300000",

currency: "SWT",

issuer: "",

freezed: "136458.0000000000",

reserve: "20",

title: "SWT"

}

],

msg: "获得余额成功",

isActive: true,

date: "Tue, 12 Nov 2019 06:32:13 GMT"

}

返回参数：

|  |  |  |  |
| --- | --- | --- | --- |
| 参数 | | 类型 | 说明 |
| result | | Boolean | 请求结果(以下所有接口返回格式都相同，不再重复描述) |
| code | | String | "0"表示正常(以下所有接口返回格式都相同，不再重复描述) |
| msg | | String | 接口返回的状态信息描述(以下所有接口返回格式都相同，不再重复描述) |
| isActive | | Boolean | 当前钱包是否激活(以下所有接口返回格式都相同，不再重复描述) |
| date | | String | 查询时间（格林威治时间）(以下所有接口返回格式都相同，不再重复描述) |
| data | | Array | 余额数组 |
|  | value | String | 余额，包含冻结金额 |
| currency | String | token |
| issuer | String | 发行方 |
| freezed | String | 冻结金额 |
| title | String | 币种名称 |
| reserve | String | 激活钱包所需币种数量 |

#### 2.3 获取交易历史记录

let res = await instance.getHistoricTransactions(address, ledger, seq)

请求参数：

address:钱包地址

ledger:账本号，首次查询可以省略，之后值为上次查询返回的ledger值

seq:上次查询的返回seq值，首次查询可以省略

结果如下：

{

"result": true,

code: "0",

data: {

marker: { ledger: 14161543, seq: 0 },

transactions: [

{

hash: "5C4A659D0EEE7A95E4CFB9F0B1C331EAD2DEFF7238D1193F2072258FA033C49D",

time: 1573174890,

type: "sell",

pairs: "CKM/SWTC",

amount: "128",

sum: "20.48",

price: "0.16",

status: "offer\_partially\_funded",

counterparty: {

account: "j4GZMwEBJot99u6d4Ywp1xvtGKQGXFMuR3",

seq: 349,

hash: "5C4A659D0EEE7A95E4CFB9F0B1C331EAD2DEFF7238D1193F2072258FA033C49D"

}

}

]

},

msg: "获取历史交易成功",

isActive: true,

date: "Tue, 12 Nov 2019 07:20:10 GMT"

}

返回参数：

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 参数 | | | | 类型 | 说明 |
| data | | | | Object |  |
|  | marker | | | Object | 记录标记 |
|  | ledger | | Number | 账本号 |
| seq | | Number | 上一次查询返回的最后一条交易号 |
| transactions | | | Array | 历史交易记录数据 |
|  |  | hash | | String | 交易hash |
| time | | Timestamp | 交易时间 |
| type | | String | 交易类型(sell/buy) |
| pairs | | String | 交易对 |
| amount | | String | 交易数量 |
|  |  | sum | | String | 总价 |
| price | | String | 单价 |
| status | | String | 交易状态(详情见下) |
| counterparty | | Object | 交易对家 |
|  | account | String | 交易对家账号 |
| seq | Number | 交易对家交易号 |
| hash | String | 交易hash |

交易状态详细说明：

offer\_funded，交易实际成交

offer\_partially\_funded，交易部分成交

offer\_cancelled，被关联交易取消单子，交易单子被取消

offer\_created，交易单子创建

offer\_bought，挂单买到/卖出，成交单子的信息

#### 2.4 获取转账历史记录

let res = await instance.getHistoricPayments(address, ledger, seq)

请求参数：

address:钱包地址

ledger:账本号，首次查询可以省略，之后值为上次查询返回的ledger值

seq:上次查询的返回seq值，首次查询可以省略

结果如下：

{

result: true,

code: "0",

data: {

marker: { ledger: 14269240, seq: 0 },

transactions: [

{

hash: "6E3A7DEEECF66F2E7CC5E1A6C4E0AC3EBE1060045D206C7897966F4B98826707",

time: 1573023290,

sender: "jKBCwv4EcyvYtD4PafP17PLpnnZ16szQsC",

receiver: "jfyiDN3XrbdPuAzWSwnx49DNsdkX6jqz12",

currency: "SWTC",

amount: "70",

result: false

}

]

},

msg: "获取历史交易成功",

isActive: true,

date: "Tue, 12 Nov 2019 08:21:26 GMT"

}

返回参数：

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 参数 | | | 类型 | 说明 |
| data | | | Array |  |
|  | marker | | Object | 记录标记 |
|  | ledger | Number | 账本号 |
| seq | Number | 交易号 |
| transactions | | Array | 转账记录数据 |
|  |  | hash | String | 转账hash |
| time | Timestamp | 转账时间 |
| sender | String | 发送方钱包地址 |
| receiver | String | 接收方钱包地址 |
| currency | String | 币种 |
| amount | String | 数量 |
| result | Boolean |  |

#### 2.5 获取当前挂单

let res = await instance.getOrders(address, page)

请求参数：

address: 钱包地址

page: 当前页

结果如下：

{

result: true,

code: "0",

data: [

{

type: "sell",

pair: "JMOAC+jGa9J9TkqtBcUoHe2zqhVFFbgUVED6o9or/CNY+jGa9J9TkqtBcUoHe2zqhVFFbgUVED6o9or",

price: "5.0000000000",

amount: "3443.0000000000",

sequence: 139,

passive: false

}

],

msg: "获得挂单列表成功",

isActive: true,

date: "Tue, 12 Nov 2019 08:43:51 GMT"

}

返回参数：

|  |  |  |  |
| --- | --- | --- | --- |
| 参数 | | 类型 | 说明 |
| data | | Array |  |
|  | type | String | 挂单类型(sell/buy) |
| pair | String | 挂单币种(挂单币种+发行方) |
| price | String | 挂单价格 |
| amoount | String | 挂单数量 |
| sequence | String | 挂单序列号 |
| passive | Boolean |  |

#### 2.6 创建挂单

let res = await instance.createOrder(sign)

请求参数：

sign: 签名

结果如下：

{

result: true,

code: "0",

data: {

hash: "CB191C4C6AE3B07A077D042CB927AAB47D63B05778389C3793220198ADBF3C27"

},

msg: "挂单提交成功",

isActive: true,

date: "Wed, 13 Nov 2019 02:46:39 GMT"

}

返回参数：

|  |  |  |  |
| --- | --- | --- | --- |
| 参数 | | 类型 | 说明 |
| data | | Object |  |
|  | hash | String | hash |

#### 2.7 取消挂单

let res = await instance.deleteOrder(sign)

请求参数：

sign: 签名

结果如下：

{

result: true,

code: "0",

data: {

hash: "2A9D8789DC22F4CE9D6A773ED6DF67DC429A456EC54A773A75D298CA551DFBE3"

},

msg: "挂单提交成功",

isActive: true,

date: "Wed, 13 Nov 2019 03:02:06 GMT"

}

结果参数：

|  |  |  |  |
| --- | --- | --- | --- |
| 参数 | | 类型 | 说明 |
| data | | Object |  |
|  | hash | String | hash |

#### 2.8 获取序列号

let res = await instance.getSequence(address)

请求参数：

address: 钱包地址

结果如下：

{

result: true,

code: "0",

data: { "sequence": 175 },

msg: "获得序列号成功",

isActive: true,

date: "Tue, 12 Nov 2019 10:10:37 GMT"

}

返回参数：

|  |  |  |
| --- | --- | --- |
| 参数 | 类型 | 说明 |
| sequence | Number | 序列号 |

#### 2.9 转账

let res = await instance.transferAccount(sign)

请求参数：

sign: 签名

结果如下：

{

result: true,

code: "0",

data: {

hash: "E29B3F74B6C22D575953924EC988B912CE2B3C7B00027BEFBA4A267A9AFF28DD"

},

msg: "提交支付成功",

isActive: true,

date: "Wed, 13 Nov 2019 03:19:43 GMT"

}

结果参数：

|  |  |  |  |
| --- | --- | --- | --- |
| 参数 | | 类型 | 说明 |
| data | | Object |  |
|  | hash | String | hash |

### 3 数据接口

#### 3.1 使用示例

const JcInfo = require("jcc\_rpc").JcInfo

let hosts = ["ijiijhg293cabc.jccdex.cn"]

let port = 443

let https = true

let instance = new JcInfo(hosts, port, https)

#### 3.2 获取指定币种24小时的行情数据

let res = await instance.getTicker(base, counter)

请求参数：

base: 基准货币

counter: 目标货币

结果如下：

{

result: true,

code: "0",

data: [

1573554754,

"0.00535",

6.786427148756566,

"0.00539",

"0.004999999999995",

1240465.8297703746,

238126229.901463,

401

],

msg: "获取市场信息成功",

success: true,

date: "Tue, 12 Nov 2019 10:32:34 GMT"

}

返回参数：

|  |  |  |  |
| --- | --- | --- | --- |
| 参数 | | 类型 | 说明 |
| data | | Array |  |
|  | data[0] | Timestamp | 当前时间 |
| data[1] | String | 当前价 |
| data[2] | Number | 日涨跌(%) |
| data[3] | String | 最高价 |
| data[4] | String | 最低价 |
| data[5] | Number | 日交易额 |
| data[6] | Number | 日成交量 |
| data[7] | Number | 成交笔数 |

#### 3.3 获取所有币种24小时的行情数据

let res = await instance.getAllTickers()

结果如下：

{

result: true,

code: "0",

data: {

VCC-SWT: [

1573556324,

"6769.889999999999",

0.00014771310568737975,

"6769.889999999999",

"6769.88",

1029022.3200000001,

152,

3

],

ECP-SWT: [1573556324, 0.45],

SWT-JUSDT: [

1573556324,

"0.000845",

0,

"0.000845",

"0.000845",

1.014000000000001,

1200,

1

]

},

msg: "获取市场信息成功",

success: true,

date: "Tue, 12 Nov 2019 10:58:44 GMT"

}

返回参数：

|  |  |  |  |
| --- | --- | --- | --- |
| 参数 | | 类型 | 说明 |
|  | Array[0] | Timestamp | 当前时间 |
| Array[1] | String | 当前价 |
| Array[2] | Number | 日涨跌(%) |
| Array[3] | String | 最高价 |
| Array[4] | String | 最低价 |
| Array[5] | Number | 日交易额 |
| Array[6] | Number | 日成交量 |
| Array[7] | Number | 成交笔数 |

#### 3.4 获取市场深度

let res = await instance.getDepth(base, counter, type)

请求参数：

base:基准货币

counter:目标货币

type: 类型 normal/more，normal数据长度为5，more数据长度为50

结果如下：

{

result: true,

code: "0",

data: {

base: "SWT",

counter: "CNY.jGa9J9TkqtBcUoHe2zqhVFFbgUVED6o9or",

asks: [

{

price: "0.00535",

amount: "1765076",

total: "1765076",

type: "sell"

}

],

bids: [

{

price: "0.00531000000000053",

amount: "376",

total: "376",

type: "buy"

}

]

},

msg: "获取市场信息成功",

success: true,

date: "Tue, 12 Nov 2019 11:34:19 GMT"

}

结果参数:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 参数 | | | 类型 | 说明 |
| data | | | Object |  |
|  | base | | String | 基准货币 |
| counter | | String | 目标货币(目标货币.发行方) |
| asks | | Array | 卖出挂单 |
|  | price | String | 价格 |
| amount | String | 数量 |
| total | String | 到当前价的总量 |
| type | String | 类型 sell:卖 buy:买 |
| bids | | Array | 买入挂单 |
|  | price | String | 价格 |
| amount | String | 数量 |
| total | String | 到当前价的总量 |
| type | String | 类型 sell:卖 buy:买 |

#### 3.5 获取K线数据

let res = await instance.getKline(base, counter, type)

请求参数：

base:基准货币

counter:目标货币

type: 类型 minute、5minute、15minute、30minute、hour、4hour、day、week、mouth

结果如下：

{

result: true,

code: '0',

data:

[

[ 1573106400000,

'0.00544',

'0.00546',

'0.00544',

'0.0055',

57663.394553305494,

10546191.052801,

19 ]

]

msg: '获取市场信息成功',

success: true,

date: 'Tue, 12 Nov 2019 12:06:02 GMT' }

结果参数：

|  |  |  |  |
| --- | --- | --- | --- |
| 参数 | | 类型 | 说明 |
| data | | Array |  |
|  | Array[0] | Timestamp | 挂单时间 |
| Array[1] | String | 开盘价 |
| Array[2] | String | 收盘价 |
| Array[3] | String | 最低价 |
| Array[4] | String | 最高价 |
| Array[5] | Number | 交易额 |
| Array[6] | Number | 交易数量 |
| Array[7] | Number | 成交笔数 |

#### 3.6 获取分时数据

let res = await instance.getHistory(base, counter, type, time)

请求参数：

base:基准货币

counter:目标货币

type: 类型 all、more、newest，如果是newest，则需要time

time: 时间戳

结果如下：

{ result: true,

code: '0',

data:

[ [ '2668.9254199999996',

'500000',

'0.00533785084',

1573561280000,

0,

1 ]

],

msg: '获取市场信息成功',

success: true,

date: 'Tue, 12 Nov 2019 12:21:55 GMT' }

结果参数：

|  |  |  |  |
| --- | --- | --- | --- |
| 参数 | | 类型 | 说明 |
| data | | Array |  |
|  | Array[0] | String | 交易额 |
| Array[1] | String | 成交量 |
| Array[2] | String | 价格 |
| Array[3] | Timestamp | 时间 |
|  | Array[4] | Number | 涨跌(0:涨，1:跌) |

#### 3.7 获取币种间汇率

let res = await instance.getTickerFromCMC(token, currency)

请求参数：

Token: 支持eth、btc

currency: 支持cny、rub

结果如下：

{ quotes: { CNY: { price: 59400 } },

last\_updated: 1570893901.299,

id: 1,

symbol: 'BTC' }

返回参数：

price: CNY与BTC之间的兑换价格(即1BTC = 58400CNY)