比特币中国 SOCKET.IO API

Socket.io API是公开使用的, [即时市场行情](http://btcchina.org/websocket-api-market-data-documentation-zh#即时市场行情)和[即时市场交易](http://btcchina.org/websocket-api-market-data-documentation-zh#即时市场交易)无需进行身份验证，但[私人订单即时更新](http://btcchina.org/websocket-api-market-data-documentation-zh#私人订单即时更新)需要使用私人秘钥进行验证。

为确保正常使用Socket.io API，请提前在您的计算机中安装*socket.io*。*Socket.io*的相关知识可参考以下网站：

<http://socket.io/>

<https://www.npmjs.org/package/socket.io>

1. 在代码中包含*socket.io*资源, 文中均为Javascript版本示例：

<script src="<YOUR PATH>/socket.io.js"></script>

2. 使用如下方式与比特币中国的websocket服务器建立连接：

var socket = io ('https://websocket.btcchina.com/');

3. 使用 “subscribe” 方法订阅我们的websocket服务器推送的不同交易市场的即时交易信息;

## 订阅三个不同的市场 ##

socket.emit('subscribe', 'marketdata\_cnybtc');

socket.emit('subscribe', 'marketdata\_cnyltc');

socket.emit('subscribe', 'marketdata\_btcltc');

socket.emit('subscribe', 'grouporder\_cnybtc');

socket.emit('subscribe', 'grouporder\_cnyltc');

socket.emit('subscribe', 'grouporder\_btcltc');

|  |  |  |
| --- | --- | --- |
| **Name** | **Type** | **Description** |
| market\_data | string | 订阅的市场数据可以是 'marketdata\_cnybtc', 'marketdata\_cnyltc' 或者 'marketdata\_btcltc'中的一个或多个，以”,”作为分隔符 |
| grouporder | string | 订阅市场深度数据，可以是 'grouporder\_cnybtc', 'grouporder\_cnyltc' 或者 'grouporder\_btcltc'中的一个或多个，以”,”作为分隔符 |

即时市场行情

使用 “ticker” 方法监听即时市场行情并处理接收到的实时数据，示例:

socket.on('ticker', function (data) { console.log(data); });

## 接收到的数据样本 ##

{

buy: 2940.03

date: 1410399073

high: 2970

last: 2940.04

low: 2901

market: "btccny"

open: 2930.15

prev\_close: 2931.03

sell: 2940.06

vol: 15187.3352

vwap: 2936.02

}

即时市场交易

使用 “trade” 方法监听即时市场交易数据并处理接收到的实时数据，示例:

socket.on('trade', function (data) { console.log(data); });

## 接收到的数据样本 ##

{

amount: 0.056

date: 1402970632

market: "btccny"

price: 3735.83

trade\_id: 6069941

type: "sell"

}

即时市场深度

即时市场深度方法会实时返回市场上的5对未成交的买卖订单。 使用 “grouporder” 方法监听即时市场深度数据并处理接收到的实时数据，示例:

socket.on('grouporder', function (data) { console.log(data); });

## 接收到的数据样本 ##

{

"grouporder":

{

"market":"btccny",

"ask":[

{"price":2405.04,"type":"ask","totalamount":0.1},

{"price":2404.24,"type":"ask","totalamount":2.9544},

{"price":2404.21,"type":"ask","totalamount":0.011},

{"price":2404.02,"type":"ask","totalamount":0.011},

{"price":2403.71,"type":"ask","totalamount":0.01}

],

"bid":[

{"price":2402.74,"type":"bid","totalamount":0.099},

{"price":2401.11,"type":"bid","totalamount":6},

{"price":2401.1,"type":"bid","totalamount":1.0014},

{"price":2400.66,"type":"bid","totalamount":0.1},

{"price":2400.65,"type":"bid","totalamount":0.1066}

]

}

}

私人订单即时更新

使用'private'方法订阅私人订单即时更新来接收处理私人订单数据。每当您的私人订单状态发生改变时，服务器就会主动推送此订单的最新状态给客户端。

此方法需要使用访问秘钥和秘密秘钥进行个人身份认证，和交易API的加密方法一致。请参照[Java示例代码](http://btcchina.org/websocket-api-market-data-documentation-zh" \l "java" \o "websocket-api-market-data-documentation-zh ↵)的以下部分进行加密和订阅处理：

*//Use 'private' method to subscribe the order feed*

[List](http://www.google.com/search?hl=en&q=allinurl%3Adocs.oracle.com+javase+docs+api+list) arg = new [ArrayList](http://www.google.com/search?hl=en&q=allinurl%3Adocs.oracle.com+javase+docs+api+arraylist)();

arg.add(sm.get\_payload());

arg.add(sm.get\_sign());

socket.emit("private",arg);

## 接收到的数据样本 ##

{

"amount": 0

"id": 3804213

"price": 33.51

"market": "ltccny"

"status": "closed"

"date":1410400468

"type":"ask"

"amount\_original":1

}

即时余额更新

使用'private'方法订阅即时余额更新来接收处理账户余额数据。每当您的余额发生改变时，服务器就会主动推送账户的相应最新余额给客户端。

此方法需要使用访问秘钥和秘密秘钥进行个人身份认证，和交易API的加密方法一致。请参照[Java示例代码](http://btcchina.org/websocket-api-market-data-documentation-zh" \l "java" \o "websocket-api-market-data-documentation-zh ↵)的以下部分进行加密和订阅处理：

*//Use 'private' method to subscribe the order and account\_info feed*

[List](http://www.google.com/search?hl=en&q=allinurl%3Adocs.oracle.com+javase+docs+api+list) arg = new [ArrayList](http://www.google.com/search?hl=en&q=allinurl%3Adocs.oracle.com+javase+docs+api+arraylist)();

arg.add(sm.get\_payload());

arg.add(sm.get\_sign());

socket.emit("private",arg);

## 接收到的数据样本 ##

{

"balance":

{

"amount\_integer":3490860,

"amount":0.0349086,

"symbol":"฿",

"amount\_decimal":8,

"currency":"BTC"

}

}

{

"balance":

{

"amount\_integer":1019107400,

"amount":10.191074,

"symbol":"¥",

"amount\_decimal":8,

"currency":"CNY"

}

}

示例代码

点击以下链接，获取更多Socket.io客户端示例源代码:

Python: <https://github.com/BTCChina/btcchina-websocket-api-python>

Java: <https://github.com/BTCChina/btcchina-websocket-api-java>

Ruby: <https://github.com/BTCChina/btcchina-websocket-api-ruby>

JavaScript: <https://github.com/BTCChina/btcchina-websocket-api-js>

C++: <https://github.com/BTCChina/btcchina-api-cpp>

C#: <https://github.com/BTCChina/btcchina-api-csharp>

JAVASCRIPT

<script src="./js/socket.io.js"></script>

<script>

var socket = io('https://websocket.btcchina.com/');

socket.emit('subscribe', 'marketdata\_cnybtc');

socket.emit('subscribe', 'marketdata\_cnyltc');

socket.emit('subscribe', 'marketdata\_btcltc');

socket.emit('subscribe', 'grouporder\_cnybtc');

socket.emit('subscribe', 'grouporder\_cnyltc');

socket.emit('subscribe', 'grouporder\_btcltc');

socket.on('connect', function(){

socket.on('trade', function (data) {

console.log(data);

});

socket.on('ticker', function (data) {

console.log(data);

});

socket.on('grouporder', function (data) {

console.log(data);

});

});

</script>

JAVA

*/\**

*\* An example for Java Socket.IO Client*

*\*/*

import *com.github.nkzawa.emitter.Emitter*;

import *com.github.nkzawa.socketio.client.IO*;

import *com.github.nkzawa.socketio.client.Socket*;

import *java.net.URISyntaxException*;

import *java.util.ArrayList*;

import *java.util.List*;

import *java.util.logging.Level*;

import *java.util.logging.Logger*;

import *javax.crypto.Mac*;

import *javax.crypto.spec.SecretKeySpec*;

import *javax.xml.bind.DatatypeConverter*;

import *org.json.JSONObject*;

public class SocketMain {

private [String](http://www.google.com/search?hl=en&q=allinurl%3Adocs.oracle.com+javase+docs+api+string) ACCESS\_KEY="YOUR\_ACCESS\_KEY";

private [String](http://www.google.com/search?hl=en&q=allinurl%3Adocs.oracle.com+javase+docs+api+string) SECRET\_KEY="YOUR\_SECRET\_KEY";

private static [String](http://www.google.com/search?hl=en&q=allinurl%3Adocs.oracle.com+javase+docs+api+string) HMAC\_SHA1\_ALGORITHM = "HmacSHA1";

private [String](http://www.google.com/search?hl=en&q=allinurl%3Adocs.oracle.com+javase+docs+api+string) postdata="";

private [String](http://www.google.com/search?hl=en&q=allinurl%3Adocs.oracle.com+javase+docs+api+string) tonce = ""+([System](http://www.google.com/search?hl=en&q=allinurl%3Adocs.oracle.com+javase+docs+api+system" \t "_blank).currentTimeMillis() \* 1000);

public static void main([String](http://www.google.com/search?hl=en&q=allinurl%3Adocs.oracle.com+javase+docs+api+string)[] args) throws [Exception](http://www.google.com/search?hl=en&q=allinurl%3Adocs.oracle.com+javase+docs+api+exception) {

try {

IO.Options opt = new IO.Options();

opt.reconnection = **true**;

Logger.getLogger(SocketMain.class.getName()).setLevel(Level.FINE);

final [Socket](http://www.google.com/search?hl=en&q=allinurl%3Adocs.oracle.com+javase+docs+api+socket) socket = IO.socket("https://websocket.btcchina.com", opt);

socket.on([Socket](http://www.google.com/search?hl=en&q=allinurl%3Adocs.oracle.com+javase+docs+api+socket" \t "_blank).EVENT\_CONNECT, new Emitter.Listener() {

SocketMain sm= new SocketMain();

@Override

public void call([Object](http://www.google.com/search?hl=en&q=allinurl%3Adocs.oracle.com+javase+docs+api+object)... args) {

[System](http://www.google.com/search?hl=en&q=allinurl%3Adocs.oracle.com+javase+docs+api+system).out.println("connected");

socket.emit("subscribe", "marketdata\_cnybtc"); *// subscribe*

socket.emit("subscribe", "marketdata\_cnyltc"); *// subscribe another market*

socket.emit("subscribe", "marketdata\_btcltc"); *// subscribe another market*

socket.emit("subscribe", "grouporder\_cnybtc"); *// subscribe grouporder*

socket.emit("subscribe", "grouporder\_cnyltc"); *// subscribe another market*

socket.emit("subscribe", "grouporder\_btcltc"); *// subscribe another market*

*//Use 'private' method to subscribe the order and account\_info feed*

try {

[List](http://www.google.com/search?hl=en&q=allinurl%3Adocs.oracle.com+javase+docs+api+list) arg = new [ArrayList](http://www.google.com/search?hl=en&q=allinurl%3Adocs.oracle.com+javase+docs+api+arraylist)();

arg.add(sm.get\_payload());

arg.add(sm.get\_sign());

socket.emit("private",arg);

} catch ([Exception](http://www.google.com/search?hl=en&q=allinurl%3Adocs.oracle.com+javase+docs+api+exception) e) {

*// TODO Auto-generated catch block*

e.printStackTrace();

}

}

}).on("trade", new Emitter.Listener() {

@Override

public void call([Object](http://www.google.com/search?hl=en&q=allinurl%3Adocs.oracle.com+javase+docs+api+object)... args) {

JSONObject json = (JSONObject) args[0]; *//receive the trade message*

[System](http://www.google.com/search?hl=en&q=allinurl%3Adocs.oracle.com+javase+docs+api+system).out.println(json);

}

}).on("ticker", new Emitter.Listener() {

@Override

public void call([Object](http://www.google.com/search?hl=en&q=allinurl%3Adocs.oracle.com+javase+docs+api+object)... args) {

JSONObject json = (JSONObject) args[0];*//receive the ticker message*

[System](http://www.google.com/search?hl=en&q=allinurl%3Adocs.oracle.com+javase+docs+api+system).out.println(json);

}

}).on("grouporder", new Emitter.Listener() {

@Override

public void call([Object](http://www.google.com/search?hl=en&q=allinurl%3Adocs.oracle.com+javase+docs+api+object)... args) {

JSONObject json = (JSONObject) args[0];*//receive the grouporder message*

[System](http://www.google.com/search?hl=en&q=allinurl%3Adocs.oracle.com+javase+docs+api+system).out.println(json);

}

}).on("order", new Emitter.Listener() {

@Override

public void call([Object](http://www.google.com/search?hl=en&q=allinurl%3Adocs.oracle.com+javase+docs+api+object)... args) {

JSONObject json = (JSONObject) args[0];*//receive your order feed*

[System](http://www.google.com/search?hl=en&q=allinurl%3Adocs.oracle.com+javase+docs+api+system).out.println(json);

}

}).on("account\_info", new Emitter.Listener() {

@Override

public void call([Object](http://www.google.com/search?hl=en&q=allinurl%3Adocs.oracle.com+javase+docs+api+object)... args) {

JSONObject json = (JSONObject) args[0];*//receive your account\_info feed*

[System](http://www.google.com/search?hl=en&q=allinurl%3Adocs.oracle.com+javase+docs+api+system).out.println(json);

}

}).on([Socket](http://www.google.com/search?hl=en&q=allinurl%3Adocs.oracle.com+javase+docs+api+socket" \t "_blank).EVENT\_DISCONNECT, new Emitter.Listener() {

@Override

public void call([Object](http://www.google.com/search?hl=en&q=allinurl%3Adocs.oracle.com+javase+docs+api+object)... args) {

[System](http://www.google.com/search?hl=en&q=allinurl%3Adocs.oracle.com+javase+docs+api+system).out.println("disconnected");

}

});

socket.connect();

} catch (URISyntaxException ex) {

Logger.getLogger(SocketMain.class.getName()).log(Level.SEVERE, **null**, ex);

}

}

public [String](http://www.google.com/search?hl=en&q=allinurl%3Adocs.oracle.com+javase+docs+api+string) get\_payload() throws [Exception](http://www.google.com/search?hl=en&q=allinurl%3Adocs.oracle.com+javase+docs+api+exception){

postdata = "{**\"**tonce**\"**:**\"**"+tonce.toString()+"**\"**,**\"**accesskey**\"**:**\"**"+ACCESS\_KEY+"**\"**,**\"**requestmethod**\"**: **\"**post**\"**,**\"**id**\"**:**\"**"+tonce.toString()+"**\"**,**\"**method**\"**: **\"**subscribe**\"**, **\"**params**\"**: [**\"**order\_cnyltc**\"**,**\"**account\_info**\"**]}";*//subscribe order feed for cnyltc market and balance feed*

[System](http://www.google.com/search?hl=en&q=allinurl%3Adocs.oracle.com+javase+docs+api+system).out.println("postdata is: " + postdata);

return postdata;

}

public [String](http://www.google.com/search?hl=en&q=allinurl%3Adocs.oracle.com+javase+docs+api+string) get\_sign() throws [Exception](http://www.google.com/search?hl=en&q=allinurl%3Adocs.oracle.com+javase+docs+api+exception){

[String](http://www.google.com/search?hl=en&q=allinurl%3Adocs.oracle.com+javase+docs+api+string) params = "tonce="+tonce.toString()+"&accesskey="+ACCESS\_KEY+"&requestmethod=post&id="+tonce.toString()+"&method=subscribe&params=order\_cnyltc,account\_info"; *//subscribe the order of cnyltc market and the account\_info*

[String](http://www.google.com/search?hl=en&q=allinurl%3Adocs.oracle.com+javase+docs+api+string) hash = getSignature(params, SECRET\_KEY);

[String](http://www.google.com/search?hl=en&q=allinurl%3Adocs.oracle.com+javase+docs+api+string) userpass = ACCESS\_KEY + ":" + hash;

[String](http://www.google.com/search?hl=en&q=allinurl%3Adocs.oracle.com+javase+docs+api+string) basicAuth = DatatypeConverter.printBase64Binary(userpass.getBytes());

return basicAuth;

}

public [String](http://www.google.com/search?hl=en&q=allinurl%3Adocs.oracle.com+javase+docs+api+string) getSignature([String](http://www.google.com/search?hl=en&q=allinurl%3Adocs.oracle.com+javase+docs+api+string) data,[String](http://www.google.com/search?hl=en&q=allinurl%3Adocs.oracle.com+javase+docs+api+string) key) throws [Exception](http://www.google.com/search?hl=en&q=allinurl%3Adocs.oracle.com+javase+docs+api+exception) {

*// get an hmac\_sha1 key from the raw key bytes*

SecretKeySpec signingKey = new SecretKeySpec(key.getBytes(), HMAC\_SHA1\_ALGORITHM);

*// get an hmac\_sha1 Mac instance and initialize with the signing key*

Mac mac = Mac.getInstance(HMAC\_SHA1\_ALGORITHM);

mac.init(signingKey);

*// compute the hmac on input data bytes*

byte[] rawHmac = mac.doFinal(data.getBytes());

return bytArrayToHex(rawHmac);

}

private [String](http://www.google.com/search?hl=en&q=allinurl%3Adocs.oracle.com+javase+docs+api+string) bytArrayToHex(byte[] a) {

StringBuilder sb = new StringBuilder();

for(byte b: a)

sb.append([String](http://www.google.com/search?hl=en&q=allinurl%3Adocs.oracle.com+javase+docs+api+string" \t "_blank).format("%02x", b&0xff));

return sb.toString();

}

}

RUBY

*# Require installing socket.io-client-simple*

require 'socket.io-client-simple'

require 'base64'

require 'json'

$access\_key = "<YOUR ACCESS KEY>"

$secret\_key = "<YOUR SECRET KEY>"

def initial\_post\_data

post\_data = {}

post\_data['tonce'] = (Time.now.to\_f \* 1000000).to\_i.to\_s

post\_data

end

def params\_string(post\_data)

post\_data['params'] = post\_data['params'].join(',')

params\_parse(post\_data).collect{|k, v| "#{k}=#{v}"} \* '&'

end

def params\_parse(post\_data)

post\_data['accesskey'] = $access\_key *#access key*

post\_data['requestmethod'] = 'post'

post\_data['id'] = post\_data['tonce'] unless post\_data.keys.include?('id')

fields=['tonce','accesskey','requestmethod','id','method','params']

ordered\_data = {}

fields.each do |field|

ordered\_data[field] = post\_data[field]

end

ordered\_data

end

def sign(params\_string)

signature = OpenSSL::HMAC.hexdigest(OpenSSL::Digest::Digest.new('sha1'), $secret\_key, params\_string) *#secret key*

Base64.strict\_encode64($access\_key+ ':' + signature)

end

socket = SocketIO::Client::Simple.connect 'https://websocket.btcchina.com'

socket.on:connect do

puts "connected!"

socket.emit **:subscribe**, "marketdata\_cnybtc"

socket.emit **:subscribe**, "marketdata\_cnyltc"

socket.emit **:subscribe**, "marketdata\_btcltc"

socket.emit **:subscribe**, "grouporder\_cnybtc"

socket.emit **:subscribe**, "grouporder\_cnyltc"

socket.emit **:subscribe**, "grouporder\_btcltc"

post\_data = initial\_post\_data

post\_data['method'] = 'subscribe'

post\_data['params'] = ["order\_cnybtc", "order\_cnyltc", "order\_btcltc", "account\_info"]

payload = params\_parse(post\_data)

pstr = params\_string(payload.clone)

signature\_string = sign(pstr)

socket.emit **:private**, [payload.to\_json, signature\_string]

end

socket.on **:disconnect** do

puts "disconnected!"

end

socket.on **:message** do |data|

puts "message: "+data

end

socket.on **:trade** do |data|

puts 'trade:'

p data

end

socket.on **:ticker** do |data|

puts 'ticker:'

p data

end

socket.on **:grouporder** do |data|

puts 'grouporder:'

p data

end

socket.on **:order** do |data|

puts 'order:'

p data

end

socket.on **:account\_info** do |data|

puts 'account\_info:'

p data

end

loop do

sleep 3

end

SOCKET.IO API V1.2.3

2014-10-27 将文档标题由Websocket API改为Socket.io API .

SOCKET.IO API V1.2.2

2014-10-15 Socket.io API v1.2.2 增加了[即时市场深度](http://btcchina.org/websocket-api-market-data-documentation-zh" \l "即时市场深度" \o "websocket-api-market-data-documentation-zh ↵)方法.

SOCKET.IO API V1.2.1

2014-09-30 Socket.io API v1.2.1 增加了[即时余额更新](http://btcchina.org/websocket-api-market-data-documentation-zh" \l "即时余额更新" \o "websocket-api-market-data-documentation-zh ↵)方法;增加了[Java的即时余额更新('account\_info')方法示例代码](http://btcchina.org/websocket-api-market-data-documentation-zh" \l "java" \o "websocket-api-market-data-documentation-zh ↵) 和 [Ruby的私人订单即时更新('order')和即时余额更新('account\_info')方法示例代码](http://btcchina.org/websocket-api-market-data-documentation-zh#ruby).

SOCKET.IO API V1.2

2014-09-11 Socket.io API v1.2 增加了[C#和C++的示例代码](http://btcchina.org/websocket-api-market-data-documentation-zh" \l "示例代码" \o "websocket-api-market-data-documentation-zh ↵); 增加了[即时市场行情](http://btcchina.org/websocket-api-market-data-documentation-zh" \l "即时市场行情" \o "websocket-api-market-data-documentation-zh ↵)和[私人订单即时更新](http://btcchina.org/websocket-api-market-data-documentation-zh#私人订单即时更新)方法;增加了[java的私人订单即时更新('order')方法示例代码](http://btcchina.org/websocket-api-market-data-documentation-zh" \l "java" \o "websocket-api-market-data-documentation-zh ↵).

SOCKET.IO API V1.1

2014-07-30 Socket.io API v1.1加入了更多的Github上的示例代码链接，包括Java，Python，Ruby和JavaScript.

SOCKET.IO API V1.0

Socket.io API v1.0 包含trade方法和JavaScript的客户端代码示例.