

# DEVELOPERS

## JSON RPC Trade API

### Content

- Trade API Authentication
- Guide to authenticate API access
  - Code Examples
  - PHP
  - Python
- Trade API v2.0.1.4
- Trade API v2.0.1.3
- Trade API v2.0.1.2
- Trade API v2.0.1.1
- Trade API v2.0.1
- Trade API v2.0
- Trade API v1.9
- Trade API v1.8
- Trade API v1.7
- Trade API v1.6
- Trade API v1.5.2
- Trade API v1.5.1
- Trade API v1.5
- Trade API v1.4
- Trade API v1.3

### Trade API Authentication#

All Trade API access is authenticated by passing the public access key and a hashed string signed private secret key. Signing is performed using Hash-based message authentication code (HMAC) from the access key and secret key, a 'tonce' is also required to be successfully authenticated. To simply the current timestamp in microseconds. This value should not differ more than 30 seconds server time. Access key and signed hash is passed using HTTP Basic Authentication method, and passed as HTTP header.

### Guide to authenticate API access#

Create the signature string to be hashed based on the following mandatory parameters. Concat key value pairs and use '&' as the separator. The order of the parameters are important. All key must be present but the value can be empty (e.g. params).

- tonce (microseconds in 16 digits, make sure your system time is synchronized with NTP. Same value can only be used once per access key.)
- accesskey (account bound and unique per user)
- requestmethod (HTTP request method, only 'post' is supported)
- id (JSON-RPC request id)
- method (JSON-RPC method name)
- params (JSON-RPC method parameters)

```
### Example 1 ###
tonce=1377743828095093
&accesskey=1d87effa-e84d-48c1-a172-0232b86305dd
&requestmethod=post
&id=1
&method=getAccountInfo
&params=

### Example 2 ###
tonce=1377743828095093
&accesskey=1d87effa-e84d-48c1-a172-0232b86305dd
&requestmethod=post
&id=1
&method=buyOrder
&params=500,1
```

Perform HMAC on the signature string using your secret key. Use sha1 as hashing algorithm.

```
### PHP ###
hash_hmac('sha1', $signature, $secretkey)
```

Utilize HTTP Basic authentication method to pass the credentials. There are two options to achieve

- Part of URI. Access key and hash can be passed directly when creating the URI request string

```
https://<accesskey>:<hash>@api.btcc.com/api_trade_v1.php
```

- HTTP Authorization header. Construct Authorization header manually and pass it along with the request. The credentials are based on access key and hash, separated by colon. The final string is base64 encoded as well



```

### PHP ###          PRODUCTS      ABOUT      NEWS (https://www.btcc.com/news)      335534676
base64_encode(<accesskey>:<hash>) //PGFjY2Vzc2tleT46PGhhc2g+
# HTTP HEADER
Authorization: Basic PGFjY2Vzc2tleT46PGhhc2g+

```

Final part is to pass 'tonce' value as HTTP header together with the request. The tonce must be i with the tonce parameter used in the signature string.

```

# HTTP HEADER
Json-Rpc-Tonce: 1377743828095093

```

Make the API request. If something is incorrect a HTTP 401 Unauthorized will be returned.

## Code Examples#

For more code samples, see the following links:

PHP: <https://github.com/BTCCChina/btcchina-api-php> (<https://github.com/BTCCChina/btcchina-api>)

Python: <https://github.com/BTCCChina/btcchina-api-python> (<https://github.com/BTCCChina/btcchina-api>)

Java: <https://github.com/BTCCChina/btcchina-api-java> (<https://github.com/BTCCChina/btcchina-api>)

.NET C#: <https://github.com/BTCCChina/btcchina-api-csharp> (<https://github.com/BTCCChina/btcchina-api>)

C++: <https://github.com/BTCCChina/btcchina-api-cpp> (<https://github.com/BTCCChina/btcchina-api>)

JavaScript: <https://github.com/BTCCChina/btcchina-api-js> (<https://github.com/BTCCChina/btcchina-api>)

## PHP#

```

<?php
function sign($method, $params = array()){
    $accessKey = "YOUR_ACCESS_KEY";
    $secretKey = "YOUR_SECRET_KEY";
    $mt = explode(' ', microtime());
    $ts = $mt[1] . substr($mt[0], 2, 6);
    $signature = urldecode(http_build_query(array(
        'tonce' => $ts,
        'accesskey' => $accessKey,
        'requestmethod' => 'post',
        'id' => 1,
        'method' => $method,
        'params' => '', //implode(',', $params),
    )));
    var_dump($signature);
    $hash = hash_hmac('sha1', $signature, $secretKey);
    return array(
        'ts' => $ts,
        'hash' => $hash,
        'auth' => base64_encode($accessKey.':'. $hash),
    );
}

function request($method, $params){
    $sign = sign($method, $params);
    $options = array(
        CURLOPT_HTTPHEADER => array(
            'Authorization: Basic ' . $sign['auth'],
            'Json-Rpc-Tonce: ' . $sign['ts'],
        ),
    );
    $postData = json_encode(array(
        'method' => $method,
        'params' => $params,
        'id' => 1,
    ));
    print($postData);
    $headers = array(
        'Authorization: Basic ' . $sign['auth'],
        'Json-Rpc-Tonce: ' . $sign['ts'],
    );
    $ch = curl_init();
    curl_setopt($ch, CURLOPT_RETURNTRANSFER, true);
    curl_setopt($ch, CURLOPT_USERAGENT,
        'Mozilla/4.0 (compatible; BTC China Trade Bot; '.php_uname('a').'; PHP/'.phpversion().')'
    );
    curl_setopt($ch, CURLOPT_URL, 'https://api.btcc.com/api_trade_v1.php');
    curl_setopt($ch, CURLOPT_POSTFIELDS, $postData);
    curl_setopt($ch, CURLOPT_HTTPHEADER, $headers);
    curl_setopt($ch, CURLOPT_SSL_VERIFYPEER, FALSE);
    // run the query
    $res = curl_exec($ch);
    return $res;
}
/**/
}
try {

```



```
var_dump(request('getAccountInfo', array()));
} catch (Exception $e) {
    echo "Error:". $e->getMessage();
}
?>
```

PRODUCTS

ABOUT

NEWS (<https://www.btcc.com/news>)

335534676

## Python#

```
#!/usr/bin/python
# -*- coding: utf-8 -*-

import btcchina

access_key="YOUR_ACCESS_KEY"
secret_key="YOUR_SECRET_KEY"

bc = btcchina.BTCChina(access_key,secret_key)

''' These methods have no arguments '''
#result = bc.get_account_info()
#print result

#result = bc.get_market_depth2()
#print result

# NOTE: for all methods shown here, the transaction ID could be set by doing
#result = bc.get_account_info(post_data={'id':'stuff'})
#print result

''' buy and sell require price (CNY, 5 decimals) and amount (LTC/BTC, 8 decimals) '''
#result = bc.buy(500,1)
#print result
#result = bc.sell(500,1)
#print result

''' cancel requires id number of order '''
#result = bc.cancel(2)
#print result

''' request withdrawal requires currency and amount '''
#result = bc.request_withdrawal('BTC',0.1)
#print result

''' get deposits requires currency. the optional "pending" defaults to true '''
#result = bc.get_deposits('BTC',pending=False)
#print result

''' get orders returns status for one order if ID is specified,
    otherwise returns all orders, the optional "open_only" defaults to true '''
#result = bc.get_orders(2)
#print result
#result = bc.get_orders(open_only=True)
#print result

''' get withdrawals returns status for one transaction if ID is specified,
    if currency is specified it returns all transactions,
    the optional "pending" defaults to true '''
#result = bc.get_withdrawals(2)
#print result
#result = bc.get_withdrawals('BTC',pending=True)
#print result

''' Fetch transactions by type. Default is 'all'.
    Available types 'all | fundbtc | withdrawbtc | fundmoney | withdrawmoney |
    refundmoney | buybtc | sellbtc | tradefee'
    Limit the number of transactions, default value is 10 '''
#result = bc.get_transactions('all',10)
#print result

'''get archived order returns a specified id order which is archived,
    the market default to "BTCCNY" and the "withdetail" default to false,if "withdetail" is spec.
    "true", the result will
    include the order's detail'''
#result = bc.get_archived_order(2,'btccny',False)
#print result

'''get archived orders returns the orders which order id is less than the specified "less_than_
d",and the returned amount
    is defined in "limit",default value is 200, if "withdetail" is specified to "true",
    the result will include to orders' detail'''
#result = bc.get_archived_orders('btccny',200,10000,False)
#print result

#!/usr/bin/env python
# -*- coding: utf-8 -*-
```



```

import time
import re
import hmac
import hashlib
import base64
import httpplib
import json

PRODUCTS      ABOUT      NEWS (https://www.btcc.com/news)  335534676

class BTCChina():
    def __init__(self, access=None, secret=None):
        self.access_key=access
        self.secret_key=secret
        self.conn=httpplib.HTTPSConnection("api.btcchina.com")

    def _get_tonce(self):
        return int(time.time()*1000000)

    def _get_params_hash(self, pdict):
        pstring=""
        # The order of params is critical for calculating a correct hash
        fields=['tonce', 'accesskey', 'requestmethod', 'id', 'method', 'params']
        for f in fields:
            if pdict[f]:
                if f == 'params':
                    # Convert list to string, then strip brackets and spaces
                    # probably a cleaner way to do this
                    param_string=re.sub("[\[\] ]","",str(pdict[f]))
                    param_string=re.sub("'",'',param_string)
                    param_string=re.sub("True",'1',param_string)
                    param_string=re.sub("False",'',param_string)
                    param_string=re.sub("None",'',param_string)
                    pstring+=f+'='+param_string+'&'
                else:
                    pstring+=f+'='+str(pdict[f])+'&'
            else:
                pstring+=f+'='
        pstring=pstring.strip('&')

        # now with correctly ordered param string, calculate hash
        phash = hmac.new(self.secret_key, pstring, hashlib.sha1).hexdigest()
        return phash

    def _private_request(self, post_data):
        #fill in common post_data parameters
        tonce=self._get_tonce()
        post_data['tonce']=tonce
        post_data['accesskey']=self.access_key
        post_data['requestmethod']='post'

        # If ID is not passed as a key of post_data, just use tonce
        if not 'id' in post_data:
            post_data['id']=tonce

        pd_hash=self._get_params_hash(post_data)

        # must use b64 encode
        auth_string='Basic '+base64.b64encode(self.access_key+'_'+pd_hash)
        headers={'Authorization':auth_string, 'Json-Rpc-Tonce':tonce}

        #post_data dictionary passed as JSON
        self.conn.request("POST", '/api_trade_v1.php', json.dumps(post_data), headers)
        response = self.conn.getresponse()

        # check response code, ID, and existence of 'result' or 'error'
        # before passing a dict of results
        if response.status == 200:
            # this might fail if non-json data is returned
            resp_dict = json.loads(response.read())

            # The id's may need to be used by the calling application,
            # but for now, check and discard from the return dict
            if str(resp_dict['id']) == str(post_data['id']):
                if 'result' in resp_dict:
                    return resp_dict['result']
                elif 'error' in resp_dict:
                    return resp_dict['error']
            else:
                # not great error handling...
                print "status:", response.status
                print "reason:", response.reason

        return None

    def get_account_info(self, post_data={}):
        post_data['method']='getAccountInfo'
        post_data['params']=[]
        return self._private_request(post_data)

```



```

def get_market_depth2(self, limit=10, market="btccny", post_data={}):
    post_data['method'] = 'getMarketDepth2'
    post_data['params'] = [limit, market]
    return self._private_request(post_data)

def buy(self, price, amount, market="btccny", post_data={}):
    amountStr = "{0:.4f}".format(round(amount,4))
    post_data['method'] = 'buyOrder2'
    if price == None:
        priceStr = None
    else:
        priceStr = "{0:.4f}".format(round(price,4))
    post_data['params'] = [priceStr, amountStr, market]
    return self._private_request(post_data)

def sell(self, price, amount, market="btccny", post_data={}):
    amountStr = "{0:.4f}".format(round(amount,4))
    post_data['method'] = 'sellOrder2'
    if price == None:
        priceStr = None
    else:
        priceStr = "{0:.4f}".format(round(price,4))
    post_data['params'] = [priceStr, amountStr, market]
    return self._private_request(post_data)

def cancel(self, order_id, market = "btccny", post_data={}):
    post_data['method'] = 'cancelOrder'
    post_data['params'] = [order_id, market]
    return self._private_request(post_data)

def request_withdrawal(self, currency, amount, post_data={}):
    post_data['method'] = 'requestWithdrawal'
    post_data['params'] = [currency, amount]
    return self._private_request(post_data)

def get_deposits(self, currency='BTC', pending=True, post_data={}):
    post_data['method'] = 'getDeposits'
    post_data['params'] = [currency, pending]
    return self._private_request(post_data)

def get_orders(self, id=None, open_only=True, market="btccny", details=True, post_data={}):
    # this combines getOrder and getOrders
    if id is None:
        post_data['method'] = 'getOrders'
        post_data['params'] = [open_only, market]
    else:
        post_data['method'] = 'getOrder'
        post_data['params'] = [id, market, details]
    return self._private_request(post_data)

def get_withdrawals(self, id='BTC', pending=True, post_data={}):
    # this combines getWithdrawal and getWithdrawals
    try:
        id = int(id)
        post_data['method'] = 'getWithdrawal'
        post_data['params'] = [id]
    except:
        post_data['method'] = 'getWithdrawals'
        post_data['params'] = [id, pending]
    return self._private_request(post_data)

def get_transactions(self, trans_type='all', limit=10, post_data={}):
    post_data['method'] = 'getTransactions'
    post_data['params'] = [trans_type, limit]
    return self._private_request(post_data)

def get_archived_order(self, id, market='btccny', withdetail=False, post_data={}):
    post_data['method'] = 'getArchivedOrder'
    post_data['params'] = [id, market, withdetail]
    return self._private_request(post_data)

def get_archived_orders(self, market='btccny', limit=200, less_than_order_id=0, withdetail=False, post_data={}):
    post_data['method'] = 'getArchivedOrders'
    post_data['params'] = [market, limit, less_than_order_id, withdetail]
    return self._private_request(post_data)

```

## JAVA

```

import javax.crypto.spec.SecretKeySpec;
import javax.crypto.Mac;
import javax.net.ssl.HttpURLConnection;
import java.io.BufferedReader;
import java.io.DataOutputStream;
import java.io.InputStreamReader;
import java.net.URL;

```



```

import java.net.URL;
import javax.xml.bind.DatatypeConverter;

class BTCCApiAuthentication {
    private static final String ACCESS_KEY = "YOUR_ACCESS_KEY";
    private static final String SECRET_KEY = "YOUR_SECRET_KEY";
    private static final String HMAC_SHA1_ALGORITHM = "HmacSHA1";

    public static String getSignature(String data, String key) throws 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 byteArrayToHex(rawHmac);
    }

    private static String byteArrayToHex(byte[] a) {
        StringBuilder sb = new StringBuilder();
        for(byte b: a)
            sb.append(String.format("%02x", b&0xff));
        return sb.toString();
    }

    public static void main(String args[]) throws Exception{
        String tonce = ""+(System.currentTimeMillis() * 1000);
        String params = "tonce="+tonce.toString()+"&accesskey="+ACCESS_KEY+"&requestmethod=post&id=
=.getAccountInfo&params=";
        String hash = getSignature(params, SECRET_KEY);
        String url = "https://api.btcc.com/api_trade_v1.php";
        URL obj = new URL(url);
        HttpURLConnection con = (HttpURLConnection) obj.openConnection();
        String userpass = ACCESS_KEY + ":" + hash;
        String basicAuth = "Basic " + DatatypeConverter.printBase64Binary(userpass.getBytes());
        //add request header
        con.setRequestMethod("POST");
        con.setRequestProperty("Json-Rpc-Tonce", tonce.toString());
        con.setRequestProperty ("Authorization", basicAuth);
        String postdata = "{\"method\": \"getAccountInfo\", \"params\": [], \"id\": 1}";
        // Send post request
        con.setDoOutput(true);
        DataOutputStream wr = new DataOutputStream(con.getOutputStream());
        wr.writeBytes(postdata);
        wr.flush();
        wr.close();
        int responseCode = con.getResponseCode();
        System.out.println("\nSending 'POST' request to URL : " + url);
        System.out.println("Post parameters : " + postdata);
        System.out.println("Response Code : " + responseCode);
        BufferedReader in = new BufferedReader(
            new InputStreamReader(con.getInputStream()));
        String inputLine;
        StringBuffer response = new StringBuffer();
        while ((inputLine = in.readLine()) != null) {
            response.append(inputLine);
        }
        in.close();
        //print result
        System.out.println(response.toString());
    }
}

```

## .NET C#

```

using System;
using System.Net;
using System.Text;
using System.Collections.Specialized;
using System.Collections.Generic;
using System.Security.Cryptography;
using System.IO;
using System.Net.Security;
using System.Security.Cryptography.X509Certificates;
namespace BTCCApiAuthentication {
    class MainClass {
        public static void Main (string[] args) {
            // For https.
            ServicePointManager.ServerCertificateValidationCallback = delegate { return true; };
            // Enter your personal API access key and secret here
            string accessKey = "YOUR_ACCESS_KEY";
            string secretKey = "YOUR_SECRET_KEY";
            string method = "getAccountInfo";
            TimeSpan timeSpan = DateTime.UtcNow - new DateTime (1970, 1, 1);
            long milliSeconds = Convert.ToInt64(timeSpan.TotalMilliseconds * 1000);
            string tonce = Convert.ToString(milliSeconds);
            NameValueCollection parameters = new NameValueCollection() {
                { "tonce", tonce },
                { "accesskey", accessKey },
                { "requestmethod", "post" },
            }
        }
    }
}

```



```

{ "id", "1" },
{ "method", method },
{ "params", "" } }, PRODUCTS ABOUT NEWS (https://www.btcc.com/news) 335534676
};
string paramsHash = GetHMACSHA1Hash(secretKey, BuildQueryString(parameters));
string base64String = Convert.ToBase64String(
Encoding.ASCII.GetBytes(accessKey + ':' + paramsHash));
string url = "https://api.btcc.com/api_trade_v1.php";
string postData = "{\n\"method\": \"\" + method + \"\", \"params\": [], \"id\": 1}";
SendPostByWebRequest(url, base64String, tonce, postData);
}
private static void SendPostByWebClient(string url, string base64,
string tonce, string postData) {
using (WebClient client = new WebClient()) {
client.Headers["Content-type"] = "application/json-rpc";
client.Headers["Authorization"] = "Basic " + base64;
client.Headers["Json-Rpc-Tonce"] = tonce;
try {
byte[] response = client.UploadData(
url, "POST", Encoding.Default.GetBytes(postData));
Console.WriteLine("\nResponse: {0}", Encoding.UTF8.GetString(response));
} catch (System.Net.WebException ex) {
Console.WriteLine(ex.Message);
}
}
}
public static void SendPostByWebRequest(string url, string base64,
string tonce, string postData) {
WebRequest webRequest = WebRequest.Create(url);
//WebRequest webRequest = HttpWebRequest.Create(url);
if (webRequest == null) {
Console.WriteLine("Failed to create web request for url: " + url);
return;
}
byte[] bytes = Encoding.ASCII.GetBytes(postData);
webRequest.Method = "POST";
webRequest.ContentType = "application/json-rpc";
webRequest.ContentLength = bytes.Length;
webRequest.Headers["Authorization"] = "Basic " + base64;
webRequest.Headers["Json-Rpc-Tonce"] = tonce;
try {
// Send the json authentication post request
using (Stream dataStream = webRequest.GetRequestStream()) {
dataStream.Write(bytes, 0, bytes.Length);
dataStream.Close();
}
// Get authentication response
using (WebResponse response = webRequest.GetResponse()) {
using (var stream = response.GetResponseStream()) {
using (var reader = new StreamReader(stream)) {
Console.WriteLine("Response: " + reader.ReadToEnd());
}
}
}
} catch (WebException ex) {
Console.WriteLine(ex.Message);
}
}
private static string BuildQueryString(NameValueCollection parameters) {
List<string> keyValues = new List<string>();
foreach (string key in parameters) {
keyValues.Add(key + "=" + parameters[key]);
}
return String.Join("&", keyValues.ToArray());
}
private static string GetHMACSHA1Hash(string secret_key, string input) {
HMACSHA1 hmacsha1 = new HMACSHA1(Encoding.ASCII.GetBytes(secret_key));
MemoryStream stream = new MemoryStream(Encoding.ASCII.GetBytes(input));
byte[] hashData = hmacsha1.ComputeHash(stream);
// Format as hexadecimal string.
StringBuilder hashBuilder = new StringBuilder();
foreach (byte data in hashData) {
hashBuilder.Append(data.ToString("x2"));
}
return hashBuilder.ToString();
}
}
}
}

```

C++

```
#include <iostream>
#include <math.h>
#include <netdb.h>
#include <openssl/ssl.h>
#include <openssl/err.h>
#include <stdio.h>
```



```
#include <string>
#include <sstream>
#include "HMAC_SHA1.h" PRODUCTS ABOUT NEWS (https://www.btcc.com/news) 335534676
using namespace std;
// Reference:
// http://www.codeproject.com/Articles/22118/C-Class-Implementation-of-HMAC-SHA
string getHmacSha1(string key, string content) {
    unsigned char digest[20];
    HMAC_SHA1 HMAC_SHA1;
    HMAC_SHA1.HMAC_SHA1((unsigned char *) content.c_str(), content.size(),
        (unsigned char *) key.c_str(), key.size(), digest);
    stringstream output;
    char result[3];
    for (int i = 0; i < 20; i++) {
        sprintf(result, "%02x", digest[i]);
        output << result;
    }
    return output.str();
}

long long getMilliseconds() {
    struct timeval start, end;
    gettimeofday( &start, NULL );
    return start.tv_sec * 1000000LL + start.tv_usec;
}

// Connects and gets the socket handle. Returns 0 if any errors.
int SocketConnect(string host_name, int port) {
    struct hostent* host = gethostbyname(host_name.c_str());
    int handle = socket(AF_INET, SOCK_STREAM, 0);
    if (handle == -1) {
        cout << "Error when creating socket to " << host_name;
        return 0;
    }
    struct sockaddr_in server;
    server.sin_family = AF_INET;
    server.sin_port = htons(port);
    server.sin_addr = *((in_addr *) host->h_addr);
    bzero(&(server.sin_zero), 8);
    int error = connect(handle, (sockaddr *) &server, sizeof(sockaddr));
    if (error == -1) {
        cout << "Error when connecting " << host_name;
        return 0;
    }
    return handle;
}

// Establishes a SSL connection. Return false if errors.
bool SslConnect(const string& server, int port,
    int& socket, SSL*& sslHandle, SSL_CTX*& sslContext) {
    socket = SocketConnect(server, port);
    if (socket == 0) {
        return false;
    }
    SSL_load_error_strings();
    SSL_library_init();
    // Use SSL 2 or 3, bind SSL to socket, and then do the SSL connection.
    if ((sslContext = SSL_CTX_new(SSLv23_client_method())) == NULL ||
        (sslHandle = SSL_new(sslContext)) == NULL ||
        SSL_set_fd(sslHandle, socket) != 1 ||
        SSL_connect(sslHandle) != 1) {
        ERR_print_errors_fp(stderr);
        return false;
    }
    return true;
}

string ReadAllFromSSL(SSL* ssl_handle) {
    const int BUCKET_READ_SIZE = 1024;
    char buffer[BUCKET_READ_SIZE];
    stringstream result;
    int received_bytes = std::numeric_limits<int>::max();
    while (received_bytes >= BUCKET_READ_SIZE) {
        received_bytes = SSL_read(ssl_handle, buffer, BUCKET_READ_SIZE);
        if (received_bytes > 0) {
            buffer[received_bytes] = '\0';
            result << buffer;
        } else {
            cout << "Error when read from SSL"; // To get the error reason, use SSL_get_error.
            return "";
        }
    }
    return result.str();
}

string Base64Encode(const string& message) {
    // Do encoding.
    BIO* bio = BIO_new(BIO_f_base64());
    BIO* bmem = BIO_new(BIO_s_mem());
    BIO_push(bio, bmem);
    BIO_set_flags(bio, BIO_FLAGS_BASE64_NO_NL); // Ignore newlines
    BIO_write(bio, message.c_str(), message.size());
    BIO_flush(bio);
}
```





```

// Get results.
char* data;
int length = BIO_get_mem_data(bmem, &data);
string encoded = string(data, length);
BIO_free_all(bio);
return encoded;
}

string GetBtcPostContent(const string& accessKey, const string& secretKey,
const string& method) {
// Get authorization token.
long long tonce = getMilliseconds();
stringstream authInput;
authInput << "tonce=" << tonce
<< "&accesskey=" << accessKey
<< "&requestmethod=post&id=1&method=" << method
<< "&params=";
string paramsHash = getHmacSha1(secretKey, authInput.str());
string authToken = Base64Encode(accessKey + ":" + paramsHash);
// Get post content.
string json_content = "{\"method\": \"getAccountInfo\", \"params\": [], \"id\": 1}";
stringstream postStream;
postStream << "POST /api_trade_v1.php HTTP/1.1\r\n"
<< "Content-Type: application/json-rpc\r\n"
<< "Authorization: Basic " << authToken << "\r\n"
<< "Json-Rpc-Tonce: " << tonce << "\r\n"
<< "Content-Length: " << json_content.size() << "\r\n"
<< "Host: api.btcc.com\r\n\r\n"
<< json_content;
string postContent = postStream.str();
cout << "POST content: " << postStream.str() << endl;
return postContent;
}

int main(int argc, char **argv) {
string accessKey = "YOUR_ACCESS_KEY";
string secretKey = "YOUR_SECRET_KEY";
string method = "getAccountInfo";
string postContent = GetBtcPostContent(accessKey, secretKey, method);
// Start the real SSL request.
int socket = 0;
SSL* sslHandle = NULL;
SSL_CTX* sslContext = NULL;
if (SslConnect("api.btcc.com", 443, socket, sslHandle, sslContext)) {
SSL_write(sslHandle, postContent.c_str(), postContent.size());
string response = ReadAllFromSSL(sslHandle);
cout << "Get response: " << response << endl;
}
// Cleanups no matter connect succeed or not.
if (socket != 0) {
close(socket);
}
if (sslHandle) {
SSL_shutdown(sslHandle);
SSL_free(sslHandle);
}
if (sslContext) {
SSL_CTX_free(sslContext);
}
return 0;
}

```

#### Trade API v2.0.1.4#

2016-01-14 Add methods for archived orders:

- getArchivedOrder
- getArchivedOrders

#### Trade API v2.0.1.3#

2014-11-19 Changed the type of "amount" and "price" from "number" to "string" in the methods buyOrder2 and sellOrder2.

#### Trade API v2.0.1.2#

2014-11-17 Added loan parameter to getAccountInfo method.

#### Trade API v2.0.1.1#

2014-10-17 Update the description of requestWithdrawal method.

#### Trade API v2.0.1#

2014-08-21 Added two more parameters for getTransactions API method: "since" and "sincetype"

#### Trade API v2.0#

2014-08-15 Released stop order API methods for all markets:

- buyStopOrder



- sellStopOrder
- getStopOrder
- getStopOrders
- cancelStopOrder

## Trade API v1.9#

2014-08-13 Added "withdetail" parameters for getOrder; added "since" and "withdetail" parameters for getOrders.

## Trade API v1.8#

2014-07-21 Added more parameters for GetAccountInfo . The parameter now could be "all", "balance", "frozen" or "profile".

## Trade API v1.7#

2014-07-17 Added Iceberg Support for all markets:

- buyIcebergOrder
- sellIcebergOrder
- getIcebergOrder
- getIcebergOrders
- cancelIcebergOrder

## Trade API v1.6#

2014-07-10 Added links for more code samples.

## Trade API v1.5.2#

2014-04-02 API updated to reduce client requests and network bandwidth.

- Updated getOrders and getMarketDepth2 to support 'market' parameter value 'ALL'.
- Added full pagination support for getOrders and getTransactions.

## Trade API v1.5.1#

2014-03-28 This release contains minor updates and bug fixes.

- Last update timestamp added in getMarketDepth2.
- Corrected the market parameter to standardized format.
- Methods getWithdrawals and getDeposits returns an empty array instead of error.
- Increased decimal places to 4 when placing BTC orders.
- Improved handling of market orders (price=null) when calling buyOrder2 and sellOrder2.

## Trade API v1.5#

2014-03-18 Release LTC/BTC market. Please use market parameter as "ltcbtc".

## Trade API v1.4#

2014-03-17 Support permission on limiting withdrawal. Permission now is 1 for Read-only, 3 for Withdraw only, and 7 for trade and withdrawal.

## Trade API v1.3#

2014-03-7 Trade API v1.3 now support LTC. This update is compatible with existing interface.

## Change log#

- Added support for LTC currency. All affected methods have a new optional currency or market parameter that can be used. BTC currency and BTC-CNY market will be used as default if the currency parameter is missing.
- Added API key permission level in getAccountInfo
- Transactions now have new types: buyltc, sellltc, fundltc, refundltc, withdrawltc. Documentation updated for existing type: rebate
- Added new statuses to withdrawal: cancel, refund, processing

## Trade API v1.2#

Trade API v1.2 is using same URL as v1, [https://api.btcc.com/api\\_trade\\_v1.php](https://api.btcc.com/api_trade_v1.php) ([https://api.btcc.com/api\\_trade\\_v1.php](https://api.btcc.com/api_trade_v1.php)).

## Change log#

Added methods buyOrder2 and sellOrder2. These will now return order id. Methods buyOrder and sellOrder are now deprecated.

## Trade API v1.1#

Trade API v1.1 is using same URL as v1, [https://api.btcc.com/api\\_trade\\_v1.php](https://api.btcc.com/api_trade_v1.php) ([https://api.btcc.com/api\\_trade\\_v1.php](https://api.btcc.com/api_trade_v1.php)).

## Change log#

- Added method getTransactions.
- Method getMarketDepth is deprecated. Use getMarketDepth2.

- Fixed parameter validation for all API methods.
- Updated error codes table.

### Trade API v1#

- Trade API is implemented based on JSON-RPC 2.0. For more information about JSON-RPC 2.0 specification please refer to their official documentation.  
http://www.jsonrpc.org/specification (http://www.jsonrpc.org/specification)
- JSON-RPC implementation for popular programming languages can be found here.  
http://json-rpc.org/wiki/implementations (http://json-rpc.org/wiki/implementations)

### Error Codes#

All API method calls will return a JSON-RPC error object if the request fails or an unexpected error occurred. **Code Message** -32000 Intern

- 32003 Insufficient CNY balance
- 32004 Insufficient BTC balance
- 32005 Order not found
- 32006 Invalid user
- 32007 Invalid currency
- 32008 Invalid amount
- 32009 Invalid wallet address
- 32010 Withdrawal not found
- 32011 Deposit not found
- 32017 Invalid type
- 32018 Invalid price
- 32019 Invalid parameter
- 32025 Order already cancelled
- 32026 Order already completed
- 32062 Lack of liquidity
- 32065 Invalid market
- 32086 Processing order

```
{
  "error":{
    "code":-32003,
    "message":"Insufficient CNY balance",
    "id": 1
  }
}
```

### Return Objects#

Successful method calls will return one of the following objects. Examples of each objects is four Method section.

### Profile#

Name	Value	Description
username	string	Account user name.
trade_password_enabled	boolean	Indicate if trade password has been set.
otp_enabled	boolean	Indicate if One Time Password (Google Authenticator) has been set.
trade_fee	number	BTC trade fee.
trade_fee_cnyltc	number	LTC trade fee.
trade_fee_btcltc	number	Swap trade fee.
daily_btc_limit	number	Maximum amount of BTC withdrawal per day.
daily_ltc_limit	number	Maximum amount of LTC withdrawal per day.
btc_deposit_address	string	Bitcoin address to deposit BTC.
btc_withdrawal_address	string	Bitcoin address to withdraw BTC.
ltc_deposit_address	string	Litecoin address to deposit LTC.
ltc_withdrawal_address	string	Litecoin address to withdraw LTC.
api_key_permission	int	Access permission for the API key. [1] Read account information. [3] Trading enabled. [5] Withdrawal only [7] Trading and Withdrawal

### Balance#

Name	Value	Description
currency	string	Currency code.
symbol	string	Currency symbol.
amount	string	Value in decimal precision.
amount_integer	string	Integer representation of the value.



amount_decimal	integer	Scale, which is number of digits to the right of the decimal of the given currency.
----------------	---------	---

## Frozen#

PRODUCTS

ABOUT

NEWS (<https://www.btcc.com/news>)

335534676

Name	Value	Description
currency	string	Currency code.
symbol	string	Currency symbol.
amount	string	Value in decimal precision.
amount_integer	string	Integer representation of the value.
amount_decimal	integer	Scale, which is number of digits to the right of the decimal of the given currency.

## Loan#

Name	Value	Description
currency	string	Currency code.
symbol	string	Currency symbol.
amount	string	Value in decimal precision.
amount_integer	string	Integer representation of the value.
amount_decimal	integer	Scale, which is number of digits to the right of the decimal of the given currency.

## Order#

Name	Value	Description
id	integer	Order id.
type	string	[ bid   ask ]
price	number	Price for 1 LTC/BTC.
currency	string	[ CNY ]
amount	number	If less than amount_original then this order has been partially filled
amount_original	number	Original amount.
date	integer	Unix time in seconds since 1 January 1970.
status	string	[ open   closed   cancelled   pending   error   insufficient_balance ]
detail	object[]	Order details, optional. Return an array of object[]: order_detail

## Withdrawal#

Name	Value	Description
id	integer	Withdrawal id.
address	string	Bitcoin/Litecoin wallet address.
currency	string	[ BTC   LTC ]
amount	number	Total amount.
date	integer	Unix time in seconds since 1 January 1970.
transacti	string	Bitcoin transaction id.
status	string	[ pending   completed   processing   cancel   refund ]

## Deposit#

Name	Value	Description
id	integer	Deposit id.
address	string	Bitcoin/Litecoin wallet address.
currency	string	[ BTC   LTC ]
amount	number	Total amount.
date	integer	Unix time in seconds since 1 January 1970.
status	string	[ pending   completed ]

## Market\_depth#

Name	Value	Description
bid	object[]	List of bid objects. Sorted by highest price first.
ask	object[]	List of ask objects. Sorted by lowest price first.

## Bid / ask#

Name	Value	Description
------	-------	-------------



price	number	Price for 1 LTC/BTC.
amount	number	Amount of LTC/BTC.

## Transaction#

Name	Value	Description
id	integer	Transaction id.
type	string	Type of transaction. 'fundbtc   withdrawbtc   fundmoney   withdrawmoney   refundmoney   buybtc   sellbtc   refundbtc   tradebtc   rebate   fundltc   refundltc   withdrawltc   buy sellltc'
btc_amount	number	Amount of BTC exchanged. Negative value indicate deduction from account.
ltc_amount	number	Amount of LTC exchanged. Negative value indicate deduction from account.
cny_amount	number	Amount of CNY exchanged. Negative value indicate deduction from account.
date	integer	Unix time in seconds since 1 January 1970.

## Iceberg\_order#

Name	Value	Description
id	integer	Iceberg order id.
type	string	[bid   ask]
price	number	Price for 1 BTC/LTC.
market	string	[BTCCNY   LTCCNY   LTCBTC]
amount	number	Amount remaining unfilled in the iceberg order.
amount_original	number	Original full amount of the iceberg order
disclosed_amount	number	The visible portion of the iceberg. This is the amount used when creating the order. This amount cannot be greater than the original amount.
variance	number	Variation of the disclosed portion so each created order is different.
date	integer	Unix time in seconds since 1 January 1970.
status	string	[ open   closed   cancelled   error ]

## Order\_detail#

Name	Value	Description
dateline	integer	Unix time in seconds since 1 January 1970.
price	number	Price for 1 BTC/LTC.
amount	number	Amount filled in this trade.

## Stop\_order#

Name	Value	Description
id	integer	stop order id.
type	string	[bid   ask]
stop_price	number	Price for 1 BTC/LTC to trigger the stop order. Can be dynamically set by the system. trailing amount/percentage is specified.
trailing_amount	number	The trailing amount to determine the stop price.
trailing_percentage	number	The trailing percentage to determine the stop price.
price	number	Price for 1 BTC/LTC for the order to be placed.
market	string	[BTCCNY   LTCCNY   LTCBTC]
amount	number	Amount used for the order to be placed.
date	integer	Create time in Unix time in seconds since 1 January 1970.
status	string	[ open   closed   cancelled   error ]
order_id	integer	Order id of the order created from this stop order, or null if stop order still open cancelled.

## Basic API Methods#

## BuyOrder#

This method is deprecated. Please use buyOrder2.

## BuyOrder2#

Place a buy order. This method will return order id.

## Parameters#

Name	Type	Required	Description
------	------	----------	-------------



price	string	YES	The price in quote currency to buy 1 base currency. Max 2 decimals for BTC/CNY and LTC/CNY markets, 4 decimals for LTC/BTC market. Market order is executed by price to 'null'.
amount	string	YES	The amount of LTC/BTC to buy. Supports 4 decimal places for BTC with a minimum amount of 0.001, and 3 decimal places for LTC.
market	string	NO	Default to "BTCCNY". [ BTCCNY   LTCCNY   LTCBTC ]

### JSON Request#

```
{
  "method": "buyOrder2",
  "params": [
    "\"500\\\", \"1\\\"",
    "id": 1
  ],
  "method": "buyOrder2",
  "params": [
    "\"500.01\\\", \"1.2312\\\", \"BTCCNY\\\"",
    "id": 1
  ],
  "method": "buyOrder2",
  "params": [
    null,
    "\"1\\\"",
    "id": 1
  ]
}
```

### Returns#

Name	Value	Description
result	integer	Returns order id if order placed successfully.

### JSON Response#

```
{
  "result": 12345,
  "id": "1"
}
```

### CancelOrder#

Cancel an active order if the status is 'open'.

### Parameters#

Name	Type	Required	Description
id	number	YES	The order id to cancel.
market	string	NO	Default to "BTCCNY". [ BTCCNY   LTCCNY   LTCBTC ]

### JSON Request#

```
{
  "method": "cancelOrder",
  "params": [
    2
  ],
  "id": 1
}
```

### Returns#

Name	Value	Description
result	boolean	Returns true if cancellation successful.

### JSON Response#

```
{
  "result": true,
  "id": "1"
}
```

### GetAccountInfo#

Get stored account information and user balance.

### Parameters#

Name	Type	Required	Description
type	string	No	Could be "all", "balance", "frozen", "loan" or "profile", default to "all".

### JSON Request#

```
{
  "method": "getAccountInfo",
  "params": [
    []
  ],
  "id": 1
},
{
  "method": "getAccountInfo",
  "params": [
    "balance"
  ],
  "id": 1
}
```

### Returns#

Name	Value	Description
result	object[]	Returns objects profile, balance, frozen and loan.

### JSON Response#

```
{
  "result": {
    "profile": {
      "username": "btc",
      "trade_password_enabled": true,
      "otp_enabled": true,
      "trade_fee": 0,

```



```

    "trade_fee_cnyltc": 0,
    "trade_fee_btcltc": 0,
    "daily_btc_limit": 10,
    "daily_ltc_limit": 300,
    "btc_deposit_address": "123myZyM9jBYGw5EB3wWmfgJ4Mvqnu7gEu",
    "btc_withdrawal_address": "123GzXJnfugniyy7ZDw3hSjkm4tHPHzHba",
    "ltc_deposit_address": "L12ysdcsNS3ZksRrVWMSohJjgcm5VQn2Tc",
    "ltc_withdrawal_address": "L23GzXJnfugniyy7ZDw3hSjkm4tHPHzHba",
    "api_key_permission": 3
  },
  "balance": {
    "btc": {
      "currency": "BTC",
      "symbol": "\u00e3f",
      "amount": "100.00000000",
      "amount_integer": "1000000000",
      "amount_decimal": 8
    },
    "ltc": {
      "currency": "LTC",
      "symbol": "\u0141",
      "amount": "0.00000000",
      "amount_integer": "0",
      "amount_decimal": 8
    },
    "cny": {
      "currency": "CNY",
      "symbol": "\u00a5",
      "amount": "50000.00000",
      "amount_integer": "5000000000",
      "amount_decimal": 5
    }
  },
  "frozen": {
    "btc": {
      "currency": "BTC",
      "symbol": "\u00e3f",
      "amount": "0.00000000",
      "amount_integer": "0",
      "amount_decimal": 8
    },
    "ltc": {
      "currency": "LTC",
      "symbol": "\u0141",
      "amount": "0.00000000",
      "amount_integer": "0",
      "amount_decimal": 8
    },
    "cny": {
      "currency": "CNY",
      "symbol": "\u00a5",
      "amount": "0.00000",
      "amount_integer": "0",
      "amount_decimal": 5
    }
  },
  "loan": {
    "btc": {
      "currency": "BTC",
      "symbol": "\u00e3f",
      "amount": "0.00000000",
      "amount_integer": "0",
      "amount_decimal": 8
    },
    "cny": {
      "currency": "CNY",
      "symbol": "\u00a5",
      "amount": "0.00000",
      "amount_integer": "0",
      "amount_decimal": 5
    }
  },
  "id": "1"
}
```

GetDeposits#

Get all user deposits.

Parameters#

Name	Type	Required	Description
currency	string	YES	[ BTC   LTC ]
pendingonly	boolean	NO	Default is 'true'. Only open deposits are returned.



## JSON Request#

PRODUCTS

ABOUT

NEWS (<https://www.btcc.com/news>)

335534676

```
{"method": "getDeposits", "params": ["BTC"], "id": 1}
```

## Returns#

Name	Value	Description
result	object[]	deposit

## JSON Response#

```
{
  "result": [
    {
      "deposit": {
        "id": 49751,
        "address": "mufUAWHCius1jZpjB4zCUwzaRbYuwXCupC",
        "currency": "BTC",
        "amount": 1,
        "date": 1376910685,
        "status": "pending"
      },
      {
        "id": 49749,
        "address": "mkrmYzYM9jBYGw5EB3wWmfgJ4Mvqnu7gEu",
        "currency": "BTC",
        "amount": 2,
        "date": 1376906645,
        "status": "completed"
      }
    ]
  }
}
```

## GetMarketDepth#

This method is deprecated. Please use `getMarketDepth2`.

## GetMarketDepth2#

Get the complete market depth. Returns all open bid and ask orders.

## Parameters#

Name	Type	Required	Description
limit	integer	NO	Limit number of orders returned. Default is 10 per side.
market	string	NO	Default to "BTCCNY". [ BTCCNY   LTCCNY   LTCBTC ]

## JSON Request#

```
{"method": "getMarketDepth2", "params": [], "id": 1}
```

## Returns#

Name	Value	Description
result	object	market_depth

## JSON Response#

```
{
  "result": {
    "market_depth": {
      "bid": [
        {
          "price": 99,
          "amount": 1
        },
        {
          "price": 98,
          "amount": 2
        }
      ],
      "ask": [
        {
          "price": 100,
          "amount": 0.997
        },
        {
          "price": 101,
          "amount": 2
        }
      ]
    }
  },
  "id": "1"
}
```

## GetOrder#

Get an order, including its status. When "withdetail" parameter is set to true, all the trade details order will be included in the response.





## Parameters#

PRODUCTS      ABOUT      NEWS (https://www.btcc.com/news)      335534676			
Name	Type	Required	Description
id	number	YES	The order id.
market	string	NO	Default to "BTCCNY". [ BTCCNY   LTCCNY   LTCBTC ]
withdetail	boolean	NO	Return the trade details or not for this order. Can be set to true, false. Default false, no detail will be returned.

## JSON Request#

```
{ "method": "getOrder", "params": [2], "id": 1 }
## fetch the order from BTCCNY market with order id=2 with order details included ##
{ "method": "getOrder", "params": [2, "BTCCNY", true], "id": 1 }
```

## Returns#

Name	Value	Description
result	object	order

## JSON Response#

```
{
  "result": {
    "order": {
      "id": 2,
      "type": "ask",
      "price": "46.84",
      "currency": "CNY",
      "amount": "0.00000000",
      "amount_original": "3.18400000",
      "date": 1406860694,
      "status": "closed",
      "details": [
        {
          "datetime": "1406860696",
          "price": "46.84",
          "amount": 3.184
        }
      ]
    },
    "id": "1"
  }
}
```

## GetOrders#

Get all orders, including status.

## Parameters#

Name	Type	Required	Description
openonly	boolean	NO	Default is 'true'. Only open orders are returned.
market	string	NO	Default to "BTCCNY". [ BTCCNY   LTCCNY   LTCBTC   ALL ]
limit	integer	NO	Limit the number of transactions, default value is 1000.
offset	integer	NO	Start index used for pagination, default value is 0.
since	integer	NO	Define the starting time from when the orders are fetched.
withdetail	boolean	NO	Return the trade details or not for this order. Can be set to true, false. Default false, no detail will be returned.

## JSON Request#

```
{ "method": "getOrders", "params": [], "id": 1 }
{ "method": "getOrders", "params": [false], "id": 1 }
## all orders from all markets limit to 2 orders per market ##
{ "method": "getOrders", "params": [false, "ALL", 2], "id": 1 }
## all orders from all markets starting from unix time 1377671475 limit to 10 filled orders per
with order details included ##
{ "method": "getOrders", "params": [true, "ALL", 10, 0, 1377671475, true], "id": 1 }
```

## Returns#

Name	Value	Description
result	object[]	order

## JSON Response#

```
## single market ##
{ "result": {
  "order": [
    {
      "id": 2,
      "type": "bid"
```



```

    "type": "bid",
    "price": 500,
    "currency": "cny",
    "amount": 0.9,
    "amount_original": 0.9,
    "date": 1377671476,
    "status": "cancelled"
  }, {
    "id": 3,
    "type": "bid",
    "price": 501,
    "currency": "cny",
    "amount": 0.8,
    "amount_original": 0.8,
    "date": 1377671475,
    "status": "cancelled"
  }
],
"id": "1"
}

## all markets ##
{
  "result": {
    "order_btccny": [
      {
        "id": 13942927,
        "type": "bid",
        "price": "2000.00",
        "currency": "CNY",
        "amount": "0.00100000",
        "amount_original": "0.00100000",
        "date": 1396255376,
        "status": "open"
      },
      {
        "id": 13942807,
        "type": "bid",
        "price": "2000.00",
        "currency": "CNY",
        "amount": "0.00100000",
        "amount_original": "0.00100000",
        "date": 1396255245,
        "status": "open"
      }
    ],
    "order_ltcnyc": [
    ],
    "order_ltcbitc": [
    ]
  },
  "id": "1"
}

```

## GetTransactions#

Get transactions log.

## Parameters#

Name	Type	Required	Description
type	string	NO	Fetch transactions by type. Default is 'all'. Available types 'all   fundbtc   withdrawbtcfec   refundbtc   fundltc   withdrawltc   withdrawltcfec   refund   fundmoney   withdrawmoney   withdrawmoneyfee   refundmoney   buybtc   buyltc   sellltc   tradefee   rebate '
limit	integer	NO	Limit the number of transactions, default value is 10.
offset	integer	NO	Start index used for pagination, default value is 0.
since	integer	NO	To fetch the transactions from this point, which can either be an transaction id or unix timestamp, default value is 0.
sincetype	string	NO	Specify the type of 'since' parameter, can either be 'id' or 'time'. default value is 'id'.

### JSON Request#

```
{ "method": "getTransactions", "params": [], "id": 1 }
{ "method": "getTransactions", "params": [ "buybtc", 2 ], "id": 1 }
# fetch 100 transactions from transaction id 101 #
{ "method": "getTransactions", "params": [ "all", 100, 0, 101, "id" ], "id": 1 }
```

## Returns#

Name	Value	Description
result	object[]	transaction



## JSON Response#

	PRODUCTS	ABOUT	NEWS (https://www.btcc.com/news)	335534676
<pre>{   "result": {     "transaction": [       {         "id": 8,         "type": "buybtc",         "btc_amount": "0.00100000",         "cny_amount": "-0.10000",         "date": 1383128749       },       {         "id": 7,         "type": "sellbtc",         "btc_amount": "-0.00100000",         "cny_amount": "0.10000",         "date": 1383128749       }     ]   },   "id": "1" }</pre>				

## GetWithdrawal#

Get withdrawal status.

## Parameters#

Name	Type	Required	Description
id	number	YES	The withdrawal id.
currency	string	NO	Default is "BTC". Can be [ BTC   LTC ]

## JSON Request#

```
{"method": "getWithdrawal", "params": [1], "id": 1}
```

## Returns#

Name	Value	Description
result	object	withdrawal

## JSON Response#

```
{
  "result": {
    "withdrawal": {
      "id": 20351,
      "address": "15MGzXJnfugniyy7ZDw3hSjkm4tHPHzHba",
      "currency": "BTC",
      "amount": 0.1,
      "date": 1376891209,
      "transaction": null,
      "status": "pending"
    }
  },
  "id": "1"
}
```

## GetWithdrawals#

Get all withdrawals.

## Parameters#

Name	Type	Required	Description
currency	string	YES	[ BTC   LTC ]
pendingonly	boolean	NO	Default is 'true'. Only pending withdrawals are returned.

## JSON Request#

```
{"method": "getWithdrawals", "params": ["BTC"], "id": 1}
```

## Returns#

Name	Value	Description
result	object[]	withdrawal

## JSON Response#

```
{
  "result": {
    "withdrawal": [
      {
        "id": 20351,
        "address": "15MGzXJnfugniyy7ZDw3hSjkm4tHPHzHba",
        "currency": "BTC",
        "amount": 0.1,
        "date": 1376891209,
        "transaction": null,
        "status": "pending"
      }
    ]
  },
  "id": "1"
}
```



```

    "id":20351,
    "address":"15MGzXJnfugniyy7ZDw3hSjkm4tHPHzHba",
    "currency":"BTC",
    "amount":0.1,
    "date":1376891209,
    "transaction":null,
    "status":"pending"
  },{
    "id":20352,
    "address":"15MGzXJnfugniyy7ZDw3hSjkm4tHPHzHba",
    "currency":"BTC",
    "amount":0.1,"date":1376891268,
    "transaction":null,
    "status":"pending"
  }
},
{"id":"1"
}

```

### RequestWithdrawal#

Make a withdrawal request. For security consideration, no BTC withdrawal address parameter is and here BTC withdrawals will pick last used withdrawal address from user profile. If the user ne change the withdrawal address, this withdrawal action should be performed manually on our w first against this new withdrawal address. NOTE: The withdrawal amount should be more than 0

### Parameters#

Name	Type	Required	Description
currency	string	YES	Currency short code, [BTC   LTC ].
amount	number	YES	Amount to withdraw.

### JSON Request#

```
{"method":"requestWithdrawal","params":["BTC",0.1],"id":1}
```

### Returns#

Name	Value	Description
result	integer	Returns the withdrawal id.

### JSON Response#

```
{"result":{"id":"20362"},"id":"1"}
```

### SellOrder#

This method is deprecated. Please use sellOrder2.

### SellOrder2#

Place a sell LTC/BTC order. This method will return order id.

### Parameters#

Name	Type	Required	Description
price	string	YES	The price in quote currency to sell 1 base currency. Max 2 decimals for BTC/CNY LTC/CNY markets. 4 decimals for LTC/BTC market. Market order is executed by s price to 'null'.
amount	string	YES	The amount of LTC/BTC to sell. Supports 4 decimal places for BTC with a minim amount of 0.001, and 3 decimal places for LTC.
market	string	NO	Default to "BTCCNY". [ BTCCNY   LTCCNY   LTCBTC ]

### JSON Request#

```

{"method":"sellOrder2","params":["500\","1\"],"id":1}
{"method":"sellOrder2","params":["500.01\","1.231\","LTCCNY\"],"id":1}
## market order ##
{"method":"sellOrder2","params":[null,"1\"],"id":1}

```

### Returns#

Name	Value	Description
result	integer	Returns order id if order placed successfully.

### JSON Response#

```
{"result":12345,"id":"1"}
```

## Iceberg Order API Methods#

PRODUCTS

ABOUT

NEWS (https://www.btcc.com/news)

335534676

An iceberg order executes a large quantity order in smaller portions for the purpose of hiding the quantity. When the visible order is completely filled, the next portion is sent to the market, until the entire iceberg order is filled. A variance can be supplied to vary the amount of the disclosed portion used to hide the iceberg order. You must have enough amount in your balance at time of the replacement of the order, but the total amount can be lower after the iceberg order is placed. If at the time a subsequent order of the disclosed amount is placed and you do not have enough in your balance, the entire order is cancelled. Any errors in placing the subsequent orders will also cancel the entire iceberg order. If you cancel or update a limit order created by the iceberg order, the entire iceberg order is cancelled.

## BuyIcebergOrder#

Place a buy iceberg order. This method will return an iceberg order id.

## Parameters#

Name	Type	Required	Description	
price	number	YES	The price in quote currency to buy 1 base currency. Max 2 decimals for BTC/CNY and LTC/CNY markets. 3 decimals for LTC/BTC market. Market iceberg order is executed by setting price to 'null'.	
amount	number	YES	The total amount of LTC/BTC to buy. Supports 3 decimal places for BTC and LTC.	
disclosed_amount	number	YES	The disclosed amount of LTC/BTC to buy. Supports 3 decimal places for BTC and LTC.	
variance	number	NO	Default to 0. Must be less than 1. When given, used as variance to the disclosed amount when the order is created.	
market	string	NO	Default to "BTCCNY". [ BTCCNY	LTCCNY

## JSON Request#

```
{ "method": "buyIcebergOrder", "params": [500, 100, 5, 0.2], "id": 1 }
## market order ##
{ "method": "buyIcebergOrder", "params": [null, 100, 5, 0.2], "id": 1 }
```

## Returns#

Name	Value	Description
result	integer	Returns iceberg order id if order placed successfully.

## #### JSON Response

```
{ "result": 12345, "id": "1" }
```

## SellIcebergOrder#

Place a sell iceberg order. This method will return an iceberg order id.

## Parameters#

Name	Type	Required	Description
price	number	YES	The price in quote currency to sell 1 base currency. Max 2 decimals for BTC/CNY and LTC/CNY markets. 3 decimals for LTC/BTC market. Market iceberg order is executed by setting price to 'null'.
amount	number	YES	The total amount of LTC/BTC to sell. Supports 3 decimal places for BTC and LTC.
disclosed_amount	number	YES	The disclosed amount of LTC/BTC to sell. Supports 3 decimal places for BTC and LTC.
variance	number	NO	Default to 0. Must be less than 1. When given, used as variance to the disclosed amount when the order is created.
market	string	NO	Default to "BTCCNY". [ BTCCNY   LTCCNY   LTCBTC ]

## JSON Request#

```
{ "method": "sellIcebergOrder", "params": [500, 100, 5, 0.2], "id": 1 }
## market order ##
{ "method": "sellIcebergOrder", "params": [null, 100, 5, 0.2], "id": 1 }
```

## Returns#



Name	Value	Description
result	integer	Returns iceberg order id if order placed successfully.

### JSON Response#

```
{"result":12345,"id":"1"}
```

### GetIcebergOrder#

Get an iceberg order, including the orders placed.

### Parameters#

Name	Type	Required	Description
id	number	YES	The iceberg order id.
market	string	NO	Default to "BTCCNY". [ BTCCNY   LTCCNY   LTCBTC ]

### JSON Request#

```
{"method":"getIcebergOrder","params":[123],"id":1}
```

### Returns#

Name	Value	Description
result	object	iceberg_order

### JSON Response#

```
{
  "result": {
    "iceberg_order": {
      "id": 1,
      "type": "bid",
      "price": "40.00",
      "market": "BTCCNY",
      "amount": "12.00000000",
      "amount_original": "12.00000000",
      "disclosed_amount": "5.00000000",
      "variance": "0.10",
      "date": 1405412126,
      "status": "open",
      "order": [
        {
          "id": 3301,
          "type": "bid",
          "price": "40.00",
          "currency": "CNY",
          "amount": "4.67700000",
          "amount_original": "4.67700000",
          "date": 1405412126,
          "status": "open"
        }
      ]
    }
  },
  "id": "1"
}
```

### GetIcebergOrders#

Get iceberg orders, including the orders placed inside each iceberg order.

### Parameters#

Name	Type	Required	Description
limit	integer	NO	Limit the number of iceberg orders, default value is 1000.
offset	integer	NO	Start index used for pagination, default value is 0.
market	string	NO	Default to "BTCCNY". [ BTCCNY   LTCCNY   LTCBTC ]

### JSON Request#

```
{"method":"getIcebergOrders","params":[],"id":1}
## 50 iceberg orders from offset at 10 ##
{"method":"getIcebergOrders","params":[50, 10],"id":1}
```

### Returns#

Name	Value	Description
result	object	iceberg_order

### JSON Response#



```
{
  "result":
    {
      "iceberg_orders":
        [
          {
            "id": 2,
            "type": "ask",
            "price": "40.00",
            "market": "BTCCNY",
            "amount": "0.00000000",
            "amount_original": "5.00000000",
            "disclosed_amount": "5.00000000",
            "variance": "0.00",
            "date": 1405412293,
            "status": "closed",
            "order":
              [
                {
                  "id": 3304,
                  "type": "ask",
                  "price": "40.00",
                  "currency": "CNY",
                  "amount": "0.00000000",
                  "amount_original": "5.00000000",
                  "date": 1405412293,
                  "status": "closed"
                }
              ]
            },
          {
            "id": 1,
            "type": "bid",
            "price": "40.00",
            "market": "BTCCNY",
            "amount": "7.00000000",
            "amount_original": "12.00000000",
            "disclosed_amount": "5.00000000",
            "variance": "0.00",
            "date": 1405412292,
            "status": "open",
            "order":
              [
                {
                  "id": 3305,
                  "type": "bid",
                  "price": "40.00",
                  "currency": "CNY",
                  "amount": "5.00000000",
                  "amount_original": "5.00000000",
                  "date": 1405412294,
                  "status": "open"
                },
                {
                  "id": 3303,
                  "type": "bid",
                  "price": "40.00",
                  "currency": "CNY",
                  "amount": "0.00000000",
                  "amount_original": "5.00000000",
                  "date": 1405412292,
                  "status": "closed"
                }
              ]
            },
          {
            "id": "1"
          }
        ]
      }
    }
  }
```

### CancelIcebergOrder#

Cancels an open iceberg order. Fails if iceberg order is already cancelled or closed. The related c with the iceberg order will also be cancelled.

#### Parameters#

Name	Type	Required	Description
id	number	YES	The iceberg order id to cancel.
market	string	NO	Default to "BTCCNY". [ BTCCNY   LTCCNY   LTCBTC ]

#### JSON Request#

```
{"method": "cancelIcebergOrder", "params": [1234], "id": 1}
```

#### Returns#

Name	Value	Description
result	boolean	Returns true if cancellation successful.

#### JSON Response#

```
{"result": true, "id": "1"}
```

### Stop Order API Methods#

A stop order allows you to put a "stop price" which will trigger an order to be placed when the st

is reached. If "price" parameter is not specified, the order placed after the stop price is reached is a market order. This type of order is also known as "stop-loss order". If "price" is set, then it will be a limit order. This type of order is also known as "stop-limit order". Note that at the point when the stop order is placed, it will not check or freeze your available balance. The amount is only checked and frozen at the point when the order is placed after the stop price is reached.

A stop trailing order allows for a stop order to have a dynamic stop price that can potentially change as the market price changes. A stop trailing order can be specified by setting the "stop trailing amount" or "stop trailing percentage", and leaving "stop price" as null. When stop trailing amount is specified, the stop price rises (in the case of sell) as the market price rises by the rising amount. (or drops in the case of buy). Similarly, if the stop trailing percentage is specified, the stop price rises (in the case of sell) as the market price rises to the trailing percentage below the market price. (or drops in the case of buy). Exactly one of the following three is specified when a stop order is created: stop price, stop trailing amount, or stop trailing percentage.

### BuyStopOrder#

Place a buy stop order. This method will return a stop order id.

#### Parameters#

Name	Type	Required	Description
stop_price	number	NO	The price in quote currency to trigger the order creation. The limit order will be triggered immediately if the last price is more than or equal to this stop price. Max 2 decimals for BTC/CNY and LTC/CNY markets, 4 decimals for LTC/BTC market. The stop price can only be specified if stop price or trailing percentage is not specified.
price	number	YES	The price in quote currency to buy 1 base currency for the order to be created when the stop price is reached. Max 2 decimals for BTC/CNY and LTC/CNY markets. 4 decimals for LTC/BTC market. Market order is created by setting price to 'null'.
amount	number	YES	The total amount of LTC/BTC to buy for the order to be created when the stop price is reached. Supports 4 decimal places for BTC with a minimum trading amount of 0.001, and 3 decimal places for LTC.
trailing_amount	number	NO	The stop trailing amount used to determine the stop price. The stop price is the trailing amount more than the lowest market price seen after the creation of the order. Can only be specified if stop price or trailing percentage is not specified.
trailing_percentage	number	NO	The stop trailing percentage used to determine the stop price. The stop price is the trailing percentage more than the lowest market price seen after the creation of the order. Can only be specified if stop price or trailing percentage is not specified.
market	string	NO	Default to "BTCCNY". [ BTCCNY   LTCCNY   LTCBTC ]

#### JSON Request#

```
## buy stop limit order ##
{"method": "buyStopOrder", "params": [500, 500.01, 0.123], "id": 1}
## buy stop market order ##
{"method": "buyStopOrder", "params": [500, null, 0.123], "id": 1}
```

#### Returns#

Name	Value	Description
result	integer	Returns stop order id if order placed successfully.

#### JSON Response#

```
{"result": 12345, "id": "1"}
```

### SellStopOrder#

Place a sell stop order. This method will return an stop order id.

#### Parameters#

Name	Type	Required	Description
stop_price	number	NO	The price in quote currency to trigger the order creation. The limit order will be triggered immediately if the last price is less than or equal to this stop price. Max 2 decimals for BTC/CNY and LTC/CNY markets, 4 decimals for LTC/BTC market. The stop price can only be specified if stop price or trailing percentage is not specified.
price	number	YES	The price in quote currency to sell 1 base currency for the order to be created when the stop price is reached. Max 2 decimals for BTC/CNY and LTC/CNY markets. 4 decimals for LTC/BTC market. Market order is created by setting price to 'null'.





amount	number	PRODUCTS	The total amount of LTC/BTC to sell for the order to be created when stop price is reached. Supports 4 decimal places for BTC and 3 decimal places for LTC.
trailing_amount	number	NO	The stop trailing amount used to determine the stop price. The stop price is the trailing amount less than the highest market price seen after the creation of the order. Can only be specified if stop price or trailing percentage is not specified.
trailing_percentage	number	NO	The stop trailing percentage used to determine the stop price. The stop price is the trailing percentage less than the highest market price seen after the creation of the order. Can only be specified if stop price or trailing percentage is not specified.
market	string	NO	Default to "BTCCNY". [ BTCCNY   LTCCNY   LTCBTC ]

### JSON Request#

```
## sell stop limit order ##
{"method":"sellStopOrder","params":[500,499.99,0.123],"id":1}
## sell stop market order ##
{"method":"sellStopOrder","params":[500,null,0.123],"id":1}
```

### Returns#

Name	Value	Description
result	integer	Returns stop order id if order placed successfully.

### JSON Response#

```
{"result":12345,"id":"1"}
```

### GetStopOrder#

Get a stop order.

### Parameters#

Name	Type	Required	Description
id	number	YES	The stop order id.
market	string	NO	Default to "BTCCNY". [ BTCCNY   LTCCNY   LTCBTC ]

### JSON Request#

```
{"method":"getStopOrder","params":[123],"id":1}
## get a stop order on LTCCNY market ##
{"method":"getStopOrder","params":[123, "LTCCNY"],"id":1}
```

### Returns#

Name	Value	Description
result	object	stop_order

### JSON Response#

```
{ "result": { "stop_order": { "id":1, "type":"bid", "stop_price":"50.00", "trailing_amount":"10.0000000",
"trailing_percentage":null, "price":"50.00", "market":"LTCCNY", "amount":"2.00000000",
"date":1407489603, "status":"open", "order_id":null}}, "id":"1" }
```

### GetStopOrders#

Get stop orders.

### Parameters#

Name	Type	Required	Description
status	string	NO	Status to filter on: [ open   closed   cancelled   error ]
type	string	NO	Type to filter on: [ ask   bid ]
stop_price	number	NO	Price to filter on. For bid stop orders, will return all stop orders less than or equal to this stop price. For ask stop orders, will return all stop orders greater than or equal to this stop price.
limit	integer	NO	Limit the number of stop orders, default value is 1000.
offset	integer	NO	Start index used for pagination, default value is 0.
market	string	NO	Default to "BTCCNY". [ BTCCNY   LTCCNY   LTCBTC ]

### JSON Request#



```
{"method":"getStopOrders","params":[],"id":1}
```

PRODUCTS

ABOUT

NEWS (<https://www.btcc.com/news>)

335534676

## Returns#

Name	Value	Description
result	object[]	Object array: stop_order

## JSON Response#

```
{
  "result":
  {
    "stop_orders":
    [
      {
        "id":1,
        "type":"bid",
        "stop_price":"50.00",
        "trailing_amount":"10.00000000",
        "trailing_percentage":null,
        "price":"50.00",
        "market":"LTCCNY",
        "amount":"2.00000000",
        "date":1407492143,
        "status":"open",
        "order_id":null,
      },
      {
        "id":2,
        "type":"bid",
        "stop_price":"44.00",
        "trailing_amount":null,
        "trailing_percentage":"0.10",
        "price":"51.00",
        "market":"LTCCNY",
        "amount":"2.00000000",
        "date":1407492144,
        "status":"open",
        "order_id":null,
      }
    ],
    "id":"1"
  }
}
```

## CancelStopOrder#

Cancels an open stop order. Fails if stop order is already cancelled or closed.

## Parameters#

Name	Type	Required	Description
id	number	YES	The stop order id to cancel.
market	string	NO	Default to "BTCCNY". [ BTCCNY   LTCCNY   LTCBTC ]

## JSON Request#

```
{"method":"cancelStopOrder","params":[123],"id":1}
## cancel a stop order on LTCCNY market ##
{"method":"cancelStopOrder","params":[123, "LTCCNY"],"id":1}
```

## Returns#

Name	Value	Description
result	boolean	Returns true if cancellation successful.

## JSON Response#

```
{"result":true,"id":"1"}
```

## getArchivedOrder#

Get an archived order. Archived order means the order which has been archived and the order wouldn't be changed.

## Parameters#

Name	Type	Required	Description
id	number	YES	The archived order id.
market	string	NO	Default to "BTCCNY". [ BTCCNY   LTCCNY   LTCBTC ]
withdetail	boolean	NO	Return the trade details or not for this archive order. Can be set to true, false. Default to false, no details will be returned.

## JSON Request#

```
{"method":"getArchivedOrder","params":[123],"id":1}
```



```
{ "method": "getArchivedOrder", "params": [2, 1, "10": 1]
```

## Returns#

PRODUCTS

ABOUT

NEWS (https://www.btcc.com/news)

335534676

Name	Value	Description
result	object	order

## JSON Response#

```
{
  "result": {
    "order": {
      "id": 2,
      "type": "ask",
      "price": "46.84",
      "currency": "CNY",
      "amount": "0.00000000",
      "amount_original": "3.18400000",
      "date": 1406860694,
      "status": "closed",
      "details": [
        {
          "datetime": "1406860696",
          "price": "46.84",
          "amount": 3.184
        }
      ]
    }
  },
  "id": "1"
}
```

## getArchivedOrders#

Get archived orders.

## Parameters#

Name	Type	Required	Description
market	string	NO	Default to "BTCCNY". [ BTCCNY   LTCCNY   LTCBTC   ALL]
limit	integer	NO	Limit the number of transactions, default value is 200.
less_than_order_id	integer	NO	Start index used for, default value is 0 which means the max order.
withdetail	boolean	NO	Return the trade details or not for orders. Can be set to true, false, to false, no details will be returned.

## JSON Request#

```
{"method": "getArchivedOrders", "params": ["BTCCNY", 10, 11, 1], "id": 1}
```

## Returns#

Name	Value	Description
result	object[]	order[]

## JSON Response#

```
{
  "result": {
    "order": [
      {
        "id": 10,
        "type": "ask",
        "price": "2.10",
        "avg_price": "431.69",
        "currency": "CNY",
        "amount": "0.00000000",
        "amount_original": "1.00000000",
        "date": 1403077028,
        "status": "closed",
        "details": [
          {
            "datetime": "1403077029",
            "price": "479.09",
            "amount": 0.4
          },
          {
            "datetime": "1403077029",
            "price": "400.09",
            "amount": 0.6
          }
        ]
      }
    ]
  }
}
```



```

    ]
  },
  {
    "id": 9,
    "type": "ask",
    "price": "2.10",
    "avg_price": 0,
    "currency": "CNY",
    "amount": "0.00000000",
    "amount_original": "1.00000000",
    "date": 1403077028,
    "status": "closed"
  }
],
"date": 1452253924
},
"date": "1"
}

```

## FAQ#

### How to solve the 401 Unauthorized problem?#

- Make sure your system time is accurate when creating the 16 digit tonce.
- Make sure your Access Key is valid and active.
- Make sure you read the Param list and use each parameter properly.
- Make sure the HTTP Authorization Header is properly sent to the BTC China server.

### How do I set parameters for getAccountInfo call?#

The input to generating the signature hash should be:

```

"  tonce=<timestamp>
  &accesskey=<yourkey>
  &requestmethod=post
  &id=1
  &method=getAccountInfo
  &params="

```

The JSON Request should be:

```

{"method": "getAccountInfo", "params": [], "id": 1}

```

### How do I set parameters for buyOrder2/sellOrder2 call?#

#### Limit order example:

The input to generating the signature hash should be:

```

"  tonce=<timestamp>
  &accesskey=<yourkey>
  &requestmethod=post
  &id=1
  &method=buyOrder2
  &params=0.0001,0.005,LTCBTC"

```

The JSON Request should be (note that you should only use up to 4 decimal for LTC/BTC for the price, and 2 decimal for the CNY price in the CNY markets):

LTC/BTC

```

{"method": "buyOrder2", "params": [0.0001, 0.005, "LTCBTC"], "id": 1}

```

LTC/CNY

```

{"method": "buyOrder2", "params": [100.00, 0.001, "LTCNY"], "id": 1}

```

BTC/CNY

```

{"method": "buyOrder2", "params": [4000.00, 0.005, "BTCNY"], "id": 1}

```

#### Market order example:

The input to generating the signature hash should be:

```

"  tonce=<timestamp>
  &accesskey=<yourkey>
  &requestmethod=post
  &id=1
  &method=buyOrder2
  &params=,0.2,ltccny"

```

The JSON Request should be (notice the "null" in the price is not the same as 0, and the signature



as empty):

```
{ "method": "buyOrder2", "params": [null, 0.2, "ltccny"], "id": 1 }
```

How to call getOrders?#

**Using openly as false example:**

The input to generating the signature hash should be:

```
"  tonce=<timestamp>
  &accesskey=<yourkey>
  &requestmethod=post
  &id=1
  &method=getOrders
  &params=false,LTCNY"
```

The JSON Request should be:

```
{ "method": "getOrders", "params": [false, "LTCNY"], "id": 1 }
```

**Using openly as true example:**

The input to generating the signature hash should be:

```
"  tonce=<timestamp>
  &accesskey=<yourkey>
  &requestmethod=post
  &id=1
  &method=getOrders
  &params=true,LTCNY"
```

The JSON Request should be :

```
{ "method": "getOrders", "params": [true, "LTCNY"], "id": 1 }
```

Note: The above example has market as "LTCNY" for the Litecoin CNY market. For other market use "BTCCNY" or "LTCBTC".

**Why do I receive 403 Forbidden?**

First, please check if you have created the key with the proper permission turned on. You can check via getAccountInfo with the key. If it does not have the proper permission, you can create a new key and add the proper permission.

**Why do I get "Invalid Amount" or "Invalid Price"?**

This happens if you specify a number too small for the market, and we round it down to 0. See the following for the amount of precision for each market:

- BTC/CNY: Use 2 decimal places for price, use 4 decimal places for amount.
- LTC/CNY: Use 2 decimal places for price, use 3 decimal places for amount.
- LTC/BTC: Use 4 decimal places for price, use 3 decimal places for amount.

**How often does Market Data API update?**

Every 5 seconds.

**When using getMarketDepth2 call, How come I receive "503 Service Temporarily Unavailable"**

We might be in the process of upgrading our servers. Please try again.

**What are the limits for using BTCC API?**

You can only call at most 5 requests per second per IP address.

**Why do I get "httpLib has no attribute HTTPSConnection" when using Python?**


Please check if you are using "import httpLib".

**Where can I download the API code examples?**

Please go to <https://github.com/BTCChina>. (<https://github.com/BTCChina>.)

**Is the price returned from getOrder my order price or filled price?**

The price returned from getOrder is the price specified in buyOrder and sellOrder. Filled price can be returned from getTransactions.



(<https://www.btcc.com>)

PRODUCTS

Bitcoin Pro Exchange  
(<https://www.btcc.com>)  
Bitcoin Pro API  
(<https://www.btcc.com>)  
Bitcoin Pro Wallet  
(<https://www.btcc.com>)  
Bitcoin Pro Mobile App  
(<https://www.btcc.com>)  
Bitcoin Pro Developer Platform  
(<https://www.btcc.com>)

ABOUT

BTCC  
(<https://www.btcc.com/about>)  
Team  
(<https://www.btcc.com/team>)  
News  
(<https://www.btcc.com/news>)  
Jobs  
(<https://www.btcc.com/jobs>)  
Contact  
(<https://www.btcc.com/contact>)  
Security  
(<https://www.btcc.com/security>)



NEWS (<https://www.btcc.com/news>)

RESOURCES

Account  
(<https://www.btcc.com/account>)  
Support  
(<https://www.btcc.com/contact>)  
Fees  
(<https://www.btcc.com/fees>)  
Privacy Policy  
(<https://www.btcc.com/legal>)

335534676

English

  
(<https://www.facebook.com/btcc>)  
(<https://twitter.com/btcc>)

Copyright © 2011 - 2016 BTCC Technology Limited