API DOCUMENTATION

BITCOIN.CO.ID

v1.7

Last updated:

6 November 2017

Table of Contents

API DOCUMENTATION FOR BITCOIN.CO.ID

Public API

These are open data for public. It doesn't need an API key to call these methods. You can call simple GET request or open it directly from the browser.

Ticker BTC/IDR - https://vip.bitcoin.co.id/api/btc_idr/ticker
Trades BTC/IDR - https://vip.bitcoin.co.id/api/btc_idr/trades
Depth BTC/IDR - https://vip.bitcoin.co.id/api/btc_idr/depth

Private API

To use Private API first you need to obtain your API credentials by logging into your Bitcoin.co.id account and open https://vip.bitcoin.co.id/trade_api. These credentials contain "API Key" and "Secret Key". Please keep these credentials safe.

There are 3 different permissions that can be applied to API Key: *view*, *trade* and *withdraw*.

Permission	**Allowed Methods**
view	*getInfo*, *transHistory*, *tradeHistory*, *openOrders*, *orderHistory*, *getOrder*
trade	*trade*, *cancelOrder*
withdraw	*withdrawCoin*

Authentication

Authorization is done by sending data via HTTP header with the following variable:

Key — API key. Example: 14B0C9D6-UG71XOID-SH4IB5VQ-A9LK3YVZ-4HFR8X2T
Sign — POST data (?param=val&param1=val1) encrypted with method HMAC-SHA512 using secret key

Send request to this URL: https://vip.bitcoin.co.id/tapi

All requests must be sent with POST.

For each request you need to include these variable to make the call valid: *nonce* and *method*.

nonce:
: An increment integer. For example if the last request's nonce is 1000, the next request should be 1001 or a larger number. To learn more about nonce http://en.wikipedia.org/wiki/Cryptographic_nonce

method:
: Specify the method you want to call.

To get the better picture on how to pass the authentication, you can check PHP function on the last page of this documentation.

Responses

All responses is returned with JSON format. See example below.

Response format if the request successfully executed:

```
{
	"success":1,
	"return":{
		/*here is some returned data*/
	}
}
```

Response format if request resulting an error:

```
{
	"success":0,
	"error":"some error message"
}
```

API Methods

getInfo

This method gives user balances and server's timestamp.

Parameter: none

Response example:

```
{
   "success":1,
   "return":{
      "balance":{
         "idr":2000000,
         "btc":1.52,
         "ltc":33.14,
         "doge":1600.299,
         "xpy":529.1239,
         ... #other coins balances
      },
      "server_time":1392225342
   }
}
```

transHistory

This method gives list of deposits and withdrawals of all currencies.

Parameter: none

Response example:

```
{
   "success":1,
   "return":{
      "withdraw":{
         "idr":[
            {
               "status":"wait",
               "type":"coupon",
               "rp":"100000",
               "fee":"0",
               "amount":"100000",
               "submit_time":"1392135074",
               "success_time":"0"
            }
         ],
         "btc":[
            {
               "status":"success",
               "btc":"150000000",
               "fee":"20000",
               "amount":"149980000",
               "submit_time":"1392135074",
               "success_time":"0"
            }
         ],
         "ltc":[
         ],
         ... #other coins
      },
      "deposit":{
         "idr":[
            {
               "status":"success",
               "type":"bank",
               "rp":"10000000",
               "fee":"0",
               "amount":"10000000",
               "submit_time":"1392193569",
               "success_time":"1392193569"
            }
         ],
         "btc":[
            {
               "status":"success",
               "btc":"200000000",
               "amount":"200000000",
               "success_time":"1391979201"
            }
         ],
         "ltc":[
         ],
         ... #other coins
      }
   }
}
```

trade

This method is for opening a new order.

Parameter:

Parameter	Required?	Description	Value	Default
pair	Yes	Pair to get the information from	btc_idr, ltc_btc, doge_btc, etc	btc_idr
type	Yes	transaction type (buy or sell)	buy / sell	-
price	Yes	order price	numerical	-
idr	required on buying btc	amount of rupiah to buy btc	numerical	-
btc	required on selling btc	amount of btc to sell	numerical	-

Response example:

```
{
   "success":1,
   "return":{
      "receive_btc":0,
      "remain_rp":1000000,
      "order_id":11560,
      "balance":{
         "idr":"8000000",
         "btc":1.52,
         "ltc":900.092,
         "doge":1552.23,
         "xpy":123.959,
         ... #other coins
      }
   }
}
```

tradeHistory

This method gives information about transaction in buying and selling history.

Parameter:

Parameter	Required?	Description	Value	Default
count	No	number of transaction which will be displayed	numerical	1000
from_id	No	first ID	numerical	0
end_id	No	end ID	numerical	∞
order	No	sort by	asc / desc	desc
since	No	start time		UNIX time
end	No	end time		UNIX time
pair	Yes	Pair to get the information from	btc_idr, ltc_btc, doge_btc, etc	btc_idr

Response example:


```
{
   "success":1,
   "return":{
      "trades":[
         {
            "trade_id":"2929",
            "order_id":"8123",
            "type":"buy",
            "btc":0.013,
            "price":"8068585",
            "fee":"1049",
            "trade_time":"1392226454"
         },
         {
            "trade_id":"2920",
            "order_id":"8111",
            "type":"sell",
            "btc":0.01499999,
            "price":"8086935",
            "fee":"1214",
            "trade_time":"1392225916"
         }
      ]
   }
}
```

openOrders

This method gives the list of current open orders (buy and sell).

Parameter:

Parameter	Required?	Description	Value	Default
pair	No	Pair to get the information from	btc_idr, ltc_btc, doge_btc, etc	-

Response example (if pair is set):

```
{
   "success":1,
   "return":{
      "orders":[
         {
            "order_id":"11567",
            "submit_time":"1392227908",
            "price":"10000000",
            "type":"buy",
            "order_idr":"1000000",
            "remain_idr":"1000000"
         }
      ]
   }
}
```

Response example (if pair is not set):

```
{
  "success": 1,
  "return": {
    "orders": {
      "btc_idr": [
        {
          "order_id": "11567",
          "submit_time": "1392227908",
          "price": "10000000",
          "type": "buy",
          "order_idr": "1000000",
          "remain_idr": "1000000"
        }
      ],
      "ltc_btc": [
        {
          "order_id": "12345",
          "submit_time": "1392228122",
          "price": "8000000",
          "type": "sell",
          "order_ltc": "100000000",
          "remain_ltc": "100000000"
        }
      ]
    }
  }
}
```

orderHistory

This method gives the list of order history (buy and sell).

Parameter:

Parameter	Required?	Description	Value	Default
pair	Yes	Pair to get the information from	btc_idr, ltc_btc, doge_btc, etc	btc_idr
count	no		integer	100
from	no		integer	0

Response example:


```
{
  "success": 1,
  "return": {
    "orders": [
      {
        "order_id": "11512",
        "type": "sell",
        "price": "5000000",
        "submit_time": "1392227908",
        "finish_time": "1392227978",
        "status": "filled",
        "order_btc": "0.00100000",
        "remain_btc": "0.00000000"
      },
      {
        "order_id": "11513",
        "type": "buy",
        "price": "5000000",
        "submit_time": "1392227908",
        "finish_time": "1392227978",
        "status": "cancelled",
        "order_idr": "1000",
        "remain_idr": "1000"
      }
    ]
  }
}
```

getOrder

Use getOrder to get specific order details.

Parameter:

Parameter	Required?	Description	Value	Default
pair	Yes	Pair to get the information from	btc_idr, ltc_btc, doge_btc, etc	btc_idr
order_id	Yes	Order ID	integer	-

Response example:


```
{
  "success": 1,
  "return": {
    "order": {
      "order_id": "94425",
      "price": "0.00810000",
      "type": "sell",
      "order_ltc": "1.00000000",
      "remain_ltc": "0.53000000",
      "submit_time": "1497657065",
      "finish_time": "0",
      "status": "open"
    }
  }
}
```



```
{
  "success": 1,
  "return": {
    "order": {
      "order_id": "664257",
      "price": "1000000000",
      "type": "buy",
      "order_rp": "10000",
      "remain_rp": "0",
      "submit_time": "1497330670",
      "finish_time": "1497330670",
      "status": "filled"
    }
  }
}
```

cancelOrder

This method is for canceling an existing open order.

Parameter:

Parameter	Required?	Description	Value	Default
pair	Yes	Pair to get the information from	btc_idr, ltc_btc, doge_btc, etc	btc_idr
order_id	Yes	Order ID	numerical	-
type	Yes	transaction type (buy or sell)	buy / sell	-

Response example:

```
{
   "success":1,
   "return":{
      "order_id":11574,
      "type":"buy",
      "balance":{
         "idr":"5000000",
         "btc":2.5,
         "ltc":900.092,
         "doge":1552.23,
         "xpy":123.959,
         ... #other coins
      }
   }
}
```

withdrawCoin

This method is for withdrawing assets (except IDR).

To be able to use this method you need to enable *withdraw* permission when you generate the API Key. Otherwise you will get “No permission” error.

You also need to prepare a *Callback URL*. *Callback URL* is a URL that our system will call to verify your withdrawal requests. Various parameters will be sent to *Callback URL*, make sure to check this information on your server side. If all the data is correct, print out a string “ok” (without quotes). We will continue the request if only we receive “ok” (without quotes) response, otherwise the request will be failed.

Callback call will be sent through a POST request, with 5 seconds connection timeout.

Request Parameter:

Parameter	Required?	Description	Value	Default
currency	Yes	Currency to withdraw	*btc*, *ltc*, *doge*, *eth*, etc	-
withdraw_address	Yes	Receiver address	a valid address	-
withdraw_amount	Yes	Amount to send	numerical	-
withdraw_memo	No	Memo to be sent to the receiver, if supported by the asset platform. Exchanges use this memo for accepting deposits for certain assets. Example: Destination Tag (for Ripple) Message (for NXT) Memo (for BitShares)	a valid memo/message/destination tag	-
request_id	Yes	Custom string you need to provide to identify each withdrawal request. request_id will be passed to callback call so your system can identify the request.	alphanumeric, max length 255	-

Response example:

```
{
  "success": 1,
  "status": "approved",
  "withdraw_currency": "xrp",
  "withdraw_address": "rwWr7KUZ3ZFwzgaDGjKBysADByzxvohQ3C",
  "withdraw_amount": "10000.00000000",
  "fee": "2.00000000",
  "amount_after_fee": "9998.00000000",
  "submit_time": "1509469200",
  "withdraw_id": "xrp-12345",
  "txid": "",
  "withdraw_memo": "123123"
}
```

Callback parameter:

Parameter	**Description**
request_id	request_id from your request
withdraw_currency	currency from your request
withdraw_address	withdraw_address from your request
withdraw_amount	withdraw_amount from your request
withdraw_memo	withdraw memo from your request (if any)
requester_ip	ip address of the request
request_date	time the request submitted

PHP Function

```
<?php

function btcid_query($method, array $req = array()) {
        // API settings
        $key = ''; // your API-key
        $secret = ''; // your Secret-key

        $req['method'] = $method;
        $req['nonce'] = time();

        // generate the POST data string
        $post_data = http_build_query($req, '', '&');

        $sign = hash_hmac('sha512', $post_data, $secret);

        // generate the extra headers
        $headers = array(
                  'Sign: '.$sign,
                  'Key: '.$key,
        );

        // our curl handle (initialize if required)
        static $ch = null;
        if (is_null($ch)) {
                $ch = curl_init();
                curl_setopt($ch, CURLOPT_RETURNTRANSFER, true);
                curl_setopt($ch, CURLOPT_USERAGENT, 'Mozilla/4.0 (compatible;
BITCOINCOID PHP client; '.php_uname('s').'; PHP/'.phpversion().')');
        }
        curl_setopt($ch, CURLOPT_URL, 'https://vip.bitcoin.co.id/tapi/');
        curl_setopt($ch, CURLOPT_POSTFIELDS, $post_data);
        curl_setopt($ch, CURLOPT_HTTPHEADER, $headers);
        curl_setopt($ch, CURLOPT_SSL_VERIFYPEER, FALSE);

        // run the query
        $res = curl_exec($ch);
        if ($res === false) throw new Exception('Could not get reply:
'.curl_error($ch));
        $dec = json_decode($res, true);
        if (!$dec) throw new Exception('Invalid data received, please make sure
connection is working and requested API exists: '.$res);

        curl_close($ch);
        $ch = null;
        return $dec;
}

$result = btcid_query('getInfo');

print_r($result);
```

Troubleshooting

I'm getting 403 Unauthorized or 403 Forbidden error.
Our firewall often too aggressive and it creates false positive for genuine requests. Please send your IP addresses to our customer service so we can manually unblock your IPs.

I'm getting “Too Many Requests” error.
To prevent abusive requests, we limit API call to 180 requests per minute.

I'm getting "Invalid nonce" error after migration to production server.
If you use timestamp for your nonce, your production server time setting might doesn't match with your development server. To prevent this you can add 86400 to the nonce. To reset nonce, disable and create a new API Key.