# Table of Contents

Public API .............................................................................................................................................3

Private API .............................................................................................................................................3

Authentication .....................................................................................................................................4
  Responses .........................................................................................................................................4

API Methods .......................................................................................................................................5
  getInfo ...............................................................................................................................................5
  transHistory .....................................................................................................................................6
  trade ..................................................................................................................................................7
  tradeHistory .....................................................................................................................................8
  openOrders .......................................................................................................................................8
  orderHistory .....................................................................................................................................10
  getOrder ..........................................................................................................................................11
  cancelOrder .....................................................................................................................................12
  withdrawCoin .................................................................................................................................13

PHP Function .......................................................................................................................................15

Troubleshooting ..................................................................................................................................16
API DOCUMENTATION FOR INDODAX.COM

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://indodax.com/api/btc_idr/ticker
Trades BTC/IDR - https://indodax.com/api/btc_idr/trades
Depth BTC/IDR - https://indodax.com/api/btc_idr/depth

Private API

To use Private API first you need to obtain your API credentials by logging into your indodax.com account and open https://indodax.com/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.

<table>
<thead>
<tr>
<th>Permission</th>
<th>Allowed Methods</th>
</tr>
</thead>
<tbody>
<tr>
<td>view</td>
<td>getInfo, transHistory, tradeHistory, openOrders, orderHistory, getOrder</td>
</tr>
<tr>
<td>trade</td>
<td>trade, cancelOrder</td>
</tr>
<tr>
<td>withdraw</td>
<td>withdrawCoin</td>
</tr>
</tbody>
</table>
**Authentication**

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

*Key* — API key. Example: 14B0C9D6-UG71XOIDS-HS4IR5VQ-A9LK3YVZ-4HFR8X2T

*Sign* — POST data (param=val&param1=val1) encrypted with method HMAC-SHA512 using secret key

Send request to this URL: [https://indodax.com/tapi](https://indodax.com/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](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:

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

Response format if request resulting an error:

```json
{
  "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:

```json
{
    "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"
                }
            ],
            ... #other coins
        }
    }
}
```
**trade**

This method is for opening a new order.

Parameter:

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

Response example:

```json
{
    "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:

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

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:
<table>
<thead>
<tr>
<th>Parameter</th>
<th>Required?</th>
<th>Description</th>
<th>Value</th>
<th>Default</th>
</tr>
</thead>
<tbody>
<tr>
<td>pair</td>
<td>No</td>
<td>Pair to get the information from</td>
<td>btc_idr, ltc_btc, doge_btc, etc</td>
<td>-</td>
</tr>
</tbody>
</table>

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:

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Required?</th>
<th>Description</th>
<th>Value</th>
<th>Default</th>
</tr>
</thead>
<tbody>
<tr>
<td>pair</td>
<td>Yes</td>
<td>Pair to get the information from</td>
<td>btc_idr, ltc_btc, doge_btc, etc</td>
<td>btc_idr</td>
</tr>
<tr>
<td>count</td>
<td>No</td>
<td>integer</td>
<td>100</td>
<td></td>
</tr>
<tr>
<td>from</td>
<td>No</td>
<td>integer</td>
<td>0</td>
<td></td>
</tr>
</tbody>
</table>

Response example:

```json
{
  "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:

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Required?</th>
<th>Description</th>
<th>Value</th>
<th>Default</th>
</tr>
</thead>
<tbody>
<tr>
<td>pair</td>
<td>Yes</td>
<td>Pair to get the information from</td>
<td><em>btc_idr, ltc_btc, doge_btc, etc</em></td>
<td><em>btc_idr</em></td>
</tr>
<tr>
<td>order_id</td>
<td>Yes</td>
<td>Order ID</td>
<td>integer</td>
<td>-</td>
</tr>
</tbody>
</table>

Response example:

```json
{
  "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"
    }
  }
}
```

```json
{
  "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:

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

Response example:

```json
{
    "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:**

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Required?</th>
<th>Description</th>
<th>Value</th>
<th>Default</th>
</tr>
</thead>
<tbody>
<tr>
<td>currency</td>
<td>Yes</td>
<td>Currency to withdraw</td>
<td><code>btc, ltc, doge, eth, etc</code></td>
<td>-</td>
</tr>
<tr>
<td>withdraw_address</td>
<td>Yes</td>
<td>Receiver address</td>
<td>a valid address</td>
<td>-</td>
</tr>
<tr>
<td>withdraw_amount</td>
<td>Yes</td>
<td>Amount to send</td>
<td>numerical</td>
<td>-</td>
</tr>
<tr>
<td>withdraw_memo</td>
<td>No</td>
<td>Memo to be sent to the receiver, if supported by the asset platform.</td>
<td>a valid memo/message/destination tag</td>
<td>-</td>
</tr>
<tr>
<td>request_id</td>
<td>Yes</td>
<td>Custom string you need to provide to identify each withdrawal request.</td>
<td>alphanumeric, max length 255</td>
<td>-</td>
</tr>
</tbody>
</table>

**Response example:**

```json
{
  "success": 1,
  "status": "approved",
  "withdraw_currency": "xrp",
  "withdraw_address": "rwWr7KUZ3ZFwzgaDGjKBysADByxzvohQ3C",
  "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:

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>request_id</td>
<td>request_id from your request</td>
</tr>
<tr>
<td>withdraw_currency</td>
<td>currency from your request</td>
</tr>
<tr>
<td>withdraw_address</td>
<td>withdraw_address from your request</td>
</tr>
<tr>
<td>withdraw_amount</td>
<td>withdraw_amount from your request</td>
</tr>
<tr>
<td>withdraw_memo</td>
<td>withdraw memo from your request (if any)</td>
</tr>
<tr>
<td>requester_ip</td>
<td>ip address of the request</td>
</tr>
<tr>
<td>request_date</td>
<td>time the request submitted</td>
</tr>
</tbody>
</table>
```php
<?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; INDODAXCOM PHP client; '.php_uname('s').'; PHP/'.phpversion().')');
    }

    curl_setopt($ch, CURLOPT_URL, 'https://indodax.com/tapi/');
    curl_setopt($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.