



# API Quick Start Guide

# Table of Contents

I.	<b>REQUEST DEPOSIT ADDRESS .....</b>	<b>2</b>
	Description	
	Example	
II.	<b>MAKE DEPOSIT .....</b>	<b>3</b>
	Description	
	Example	
III.	<b>QUERY TOKEN BALANCE .....</b>	<b>4</b>
	Description	
	Example	
IV.	<b>VALIDATE RECEIVER ADDRESS .....</b>	<b>5</b>
	Description	
	Example	
V.	<b>REQUEST WITHDRAWAL .....</b>	<b>6</b>
	Description	
	Example	
VI.	<b>QUERY ORDER .....</b>	<b>7</b>
	Description	
	Example	
VII.	<b>REQUEST AUDIT .....</b>	<b>8</b>
	Description	
	Example	
VIII.	<b>QUERY AUDIT .....</b>	<b>9</b>
	Description	
	Example	

# REQUEST DEPOSIT ADDRESS

## Description

Deposit is one of main services provided by Jadepool. End user needs to request a deposit address through the client system and transfers asset from his / her own wallet to the address.

In order to support one blockchain, Jadepool must be configured with at least one node. With the node in sync with the chain, a scheduled task, whose interval settings can be customized, starts to scan all mined transactions in the new block and save any as “order” to database if the transaction should be accounted for. Considering the high possibility of temporary soft fork, every blockchain supported by Jadepool is configured with a “soft fork gap” setting in case Jadepool falsely saves transactions from invalid blocks, that means Jadepool only scans the block whose height is below the current block header minus the “soft fork gap”. Once the transaction is saved to database, Jadepool will keep track of it by querying transaction hash from node until it reaches the final state. No matter “done” or failed, Jadepool will make sure that the client system is informed.

## Example

The screenshot shows a REST client interface with a POST request to `http://localhost:7001/api/v1/addresses/new`. The response is displayed in JSON format, showing a successful deposit address generation. The `address` field is highlighted with a red box.

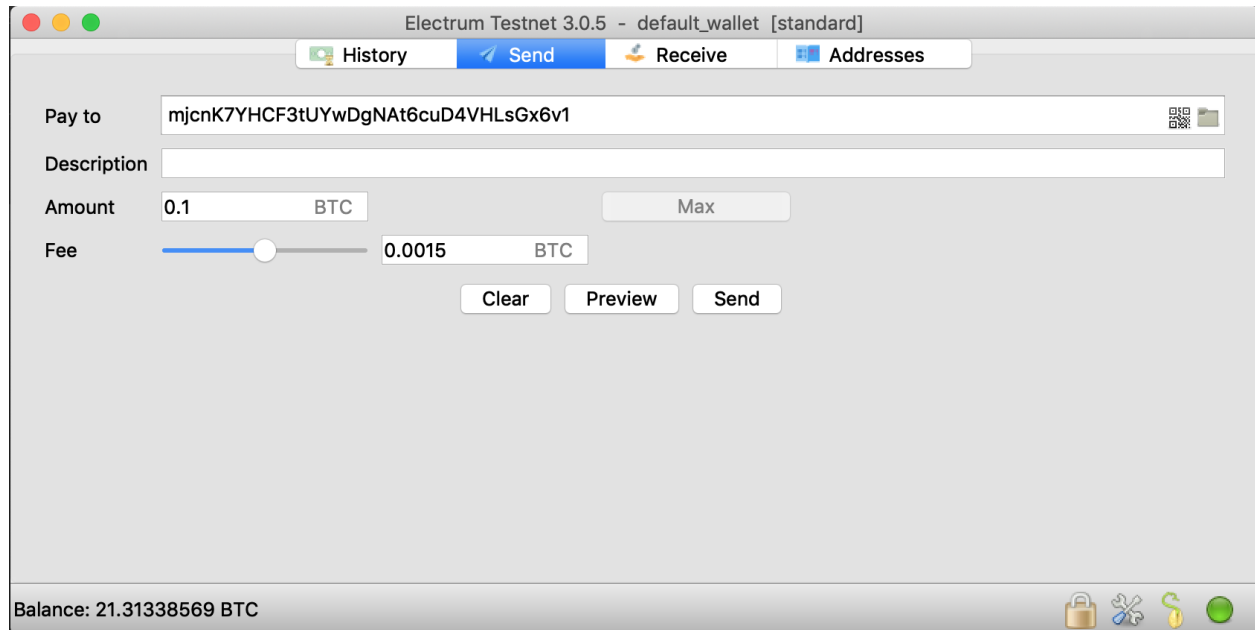
```
1 {
2   "code": 0,
3   "status": 0,
4   "message": "OK",
5   "crypto": "ecc",
6   "timestamp": 1548316374230,
7   "sig": {
8     "r": "LvGXqrI0eaB0A/SUvvgZuNerIcUjGHLqvApZGkLuZ/Y=",
9     "s": "cYmP5ixfL3mwY2G4hLoNSUAeIfLP9YuImBu27/YtrCc=",
10    "v": 27
11  },
12  "result": {
13    "type": "BTC",
14    "subType": "BTC",
15    "address": "mjcnK7YHCF3tUYwDgNA6cuD4VHLsGx6v1",
16    "namespace": "BTC",
17    "sid": "fXvlt-ZX9vLn3jAmAAAC"
18  }
19 }
```

# MAKE DEPOSIT

## Description

Once obtained a deposit address from the step above, make a transfer to the address from your own wallet. The example below is using Bitcoin Electrum. Please refer to “How To Install & Use Wallets” documents for detailed guide of wallet installation and operation.

## Example



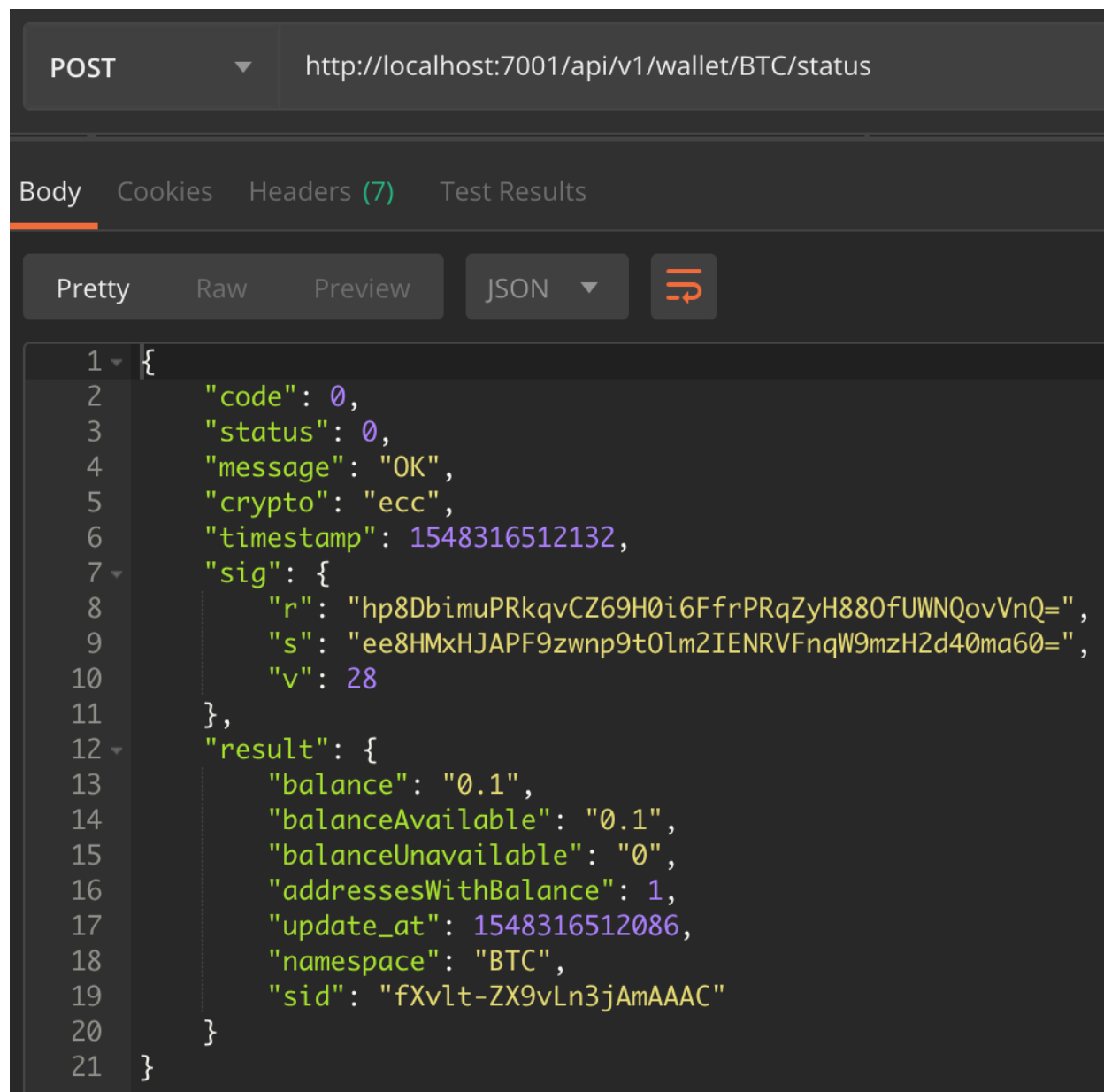
# QUERY TOKEN BALANCE

## Description

After making transfer to the deposit address, it should be shown in the balance if the transaction is broadcasted successfully.

“balance” is the total balance of the token in Jadepool. “balanceAvailable” is the value that can be used for outcoming transfer (withdrawal and “hot to cold” etc.).

## Example



The screenshot shows a REST client interface with a POST request to `http://localhost:7001/api/v1/wallet/BTC/status`. The response is displayed in JSON format, showing a successful status with a balance of 0.1.

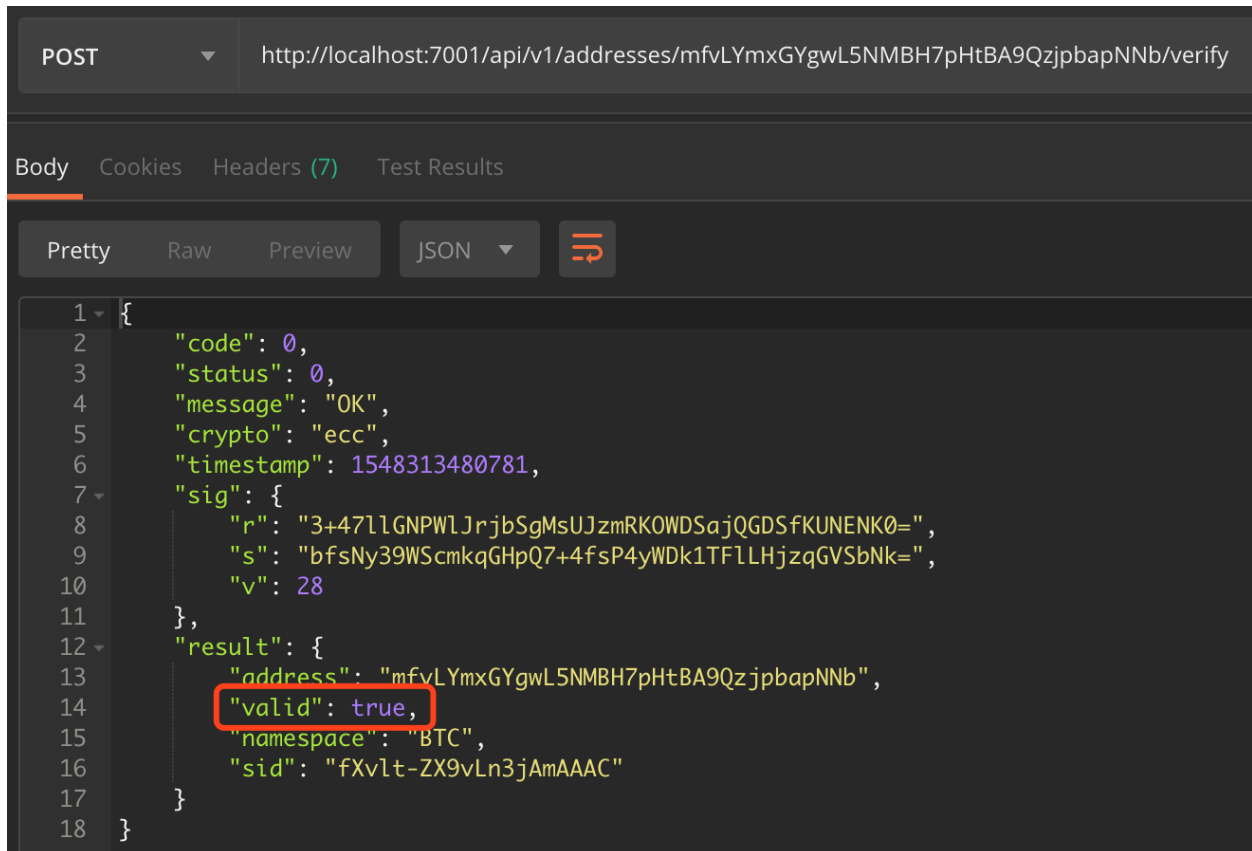
```
1 {
2   "code": 0,
3   "status": 0,
4   "message": "OK",
5   "crypto": "ecc",
6   "timestamp": 1548316512132,
7   "sig": {
8     "r": "hp8DbimuPRkqvCZ69H0i6FfrPRqZyH880fUWNQovVnQ=",
9     "s": "ee8HMxHJAPF9zwnp9t0lm2IENRVFnqW9mzH2d40ma60=",
10    "v": 28
11  },
12  "result": {
13    "balance": "0.1",
14    "balanceAvailable": "0.1",
15    "balanceUnavailable": "0",
16    "addressesWithBalance": 1,
17    "update_at": 1548316512086,
18    "namespace": "BTC",
19    "sid": "fXvlt-ZX9vLn3jAmAAAC"
20  }
21 }
```

# VALIDATE RECEIVER ADDRESS

## Description

Now Jadepool has balance so it can process withdrawals. Before request a withdrawal, make sure to validate the receiver address first so the withdrawal won't fail on this reason.

## Example



```
POST http://localhost:7001/api/v1/addresses/mfvLYmxGYgwL5NMBH7pHtBA9QzjbapNNb/verify

Body Cookies Headers (7) Test Results

Pretty Raw Preview JSON

1 {
2   "code": 0,
3   "status": 0,
4   "message": "OK",
5   "crypto": "ecc",
6   "timestamp": 1548313480781,
7   "sig": {
8     "r": "3+4711GNPWLJrjbSgMsUJzmRKOWDSajQGDSfkUNENK0=",
9     "s": "bfsNy39WScmkqGHpQ7+4fsP4yWDk1TF1LHjzqGVSbNk=",
10    "v": 28
11  },
12  "result": {
13    "address": "mfvLYmxGYgwL5NMBH7pHtBA9QzjbapNNb",
14    "valid": true,
15    "namespace": "BTC",
16    "sid": "fXvlt-ZX9vLn3jAmAAAC"
17  }
18 }
```

# REQUEST WITHDRAWAL

## Description

After the receiver address is validated, request a withdrawal and obtain the withdrawal order ID. The example below has the order ID of 6413.

## Example

```
POST http://127.0.0.1:7001/api/v1/transactions/

Pretty Raw Preview JSON

{
  "code": 0,
  "status": 0,
  "message": "OK",
  "crypto": "ecc",
  "timestamp": 1548316656549,
  "sig": {
    "r": "JPDzy0bUgN1ozx9P8ds02W9VpAQcsUL85N4/tPNW0Qw=",
    "s": "b5EMFd+D5M1+wKeIuXaTa5ND4r8mhrp9aQkSfYZlQ+g=",
    "v": 28
  },
  "result": {
    "data": {},
    "id": "6413",
    "state": "init",
    "bizType": "WITHDRAW",
    "type": "BTC",
    "coinType": "BTC",
    "to": "mfvLYmxGYgwL5NMBH7pHtBA9QzjpbapNNb",
    "value": "0.08",
    "confirmations": 0,
    "create_at": 1548316656499,
    "update_at": 1548316656500,
    "from": "myKRmYjEzPUtsqSUzy6yMEmpMT491Rj6Va",
    "fee": "0",
    "hash": "",
    "extraData": "",
    "memo": "",
    "sendAgain": false,
    "namespace": "BTC",
    "sid": "fXvlt-ZX9vLn3jAmAAAC"
  }
}
```

# QUERY ORDER

## Description

Key information can be obtained from querying the order with the order ID from the step above, including the transaction hash, fee and the current status.

## Example

```
GET http://localhost:7001/api/v1/transactions/6413

{
  "result": {
    "id": "6413",
    "state": "pending",
    "bizType": "WITHDRAW",
    "type": "BTC",
    "coinType": "BTC",
    "to": "mfvLYmxGYgwL5NMBH7pHtBA9QzjpbapNNb",
    "value": "0.08",
    "confirmations": 1,
    "create_at": 1548316656499,
    "update_at": 1548316679810,
    "from": "mjcnK7YHCF3tUYwDgNAt6cuD4VHLsGx6v1",
    "fee": "0.00109375",
    "data": {
      "timestampBegin": 1548316657778,
      "timestampFinish": 0,
      "timestampHandle": 1548316657572,
      "type": "Bitcoin",
      "hash": "4427227cb68165f190342f857ba1a1871654d170e732aa731fbd184990be058a",
      "fee": 0.00109375,
      "blockNumber": 1453798,
      "blockHash": "00000000000000a99537182e1d22369cbadf39d82901d2baf18f74f204aaf1e7",
      "confirmations": 1,
      "from": [
        {
          "address": "mjcnK7YHCF3tUYwDgNAt6cuD4VHLsGx6v1",
          "value": "0.1",
          "txid": "1871c7dc9a1c62511c53f4aafcc7fa8cd1eddb1ed40d531fcf03a74378ac5059",
          "n": 0
        }
      ],
      "to": [
        {
          "address": "mfvLYmxGYgwL5NMBH7pHtBA9QzjpbapNNb",
          "value": "0.08000000",
          "txid": "",
          "n": 0
        },
        {
          "address": "myKRmYjEzPUTsqSUzy6yMEmpMT491Rj6Va",
          "value": "0.01890625",
          "txid": "",
          "n": 1
        }
      ],
      "state": "pending"
    },
    "hash": "4427227cb68165f190342f857ba1a1871654d170e732aa731fbd184990be058a"
  }
}
```



# REQUEST AUDIT

## Description

To request an audit, JadePool finds the nearest block around the timestamp given in API call and then do calculation on all orders between the block from the last audit and the current. If request audit for the first time, all orders in database will be calculated. Please note, audit will only calculate orders whose state is final, and the timestamp given in API call must be later than the creation timestamp of the lowest block height saved in JadePool database, and it must not be later than the current timestamp.

## Example

The screenshot shows a REST client interface with a POST request to `localhost:7001/api/v1/audits`. The response is displayed in JSON format. The JSON object contains a `code` of 0, a `status` of 0, a `message` of "OK", a `crypto` of "ecc", and a `timestamp` of 1548321439086. It also includes a `sig` object with `r`, `s`, and `v` values. The `result` object contains `type` "BTC", a `current` object with `id` "5c49829fdddec563c50f389", `type` "BTC", `blocknumber` 1453802, and `timestamp` 1548321429000. Other fields include `last` (null), `namespace` "BTC", and `sid` "fRrwinMmWUa29PlrAAAB".

```
{
  "code": 0,
  "status": 0,
  "message": "OK",
  "crypto": "ecc",
  "timestamp": 1548321439086,
  "sig": {
    "r": "cWQl45F0hwQJgvppQKwPbV5FbDUDL/MIstazcmzos2g=",
    "s": "USlHjoQfy65wLVl4YzIJmqzKdZMwBdqAe4kI9LfTc44=",
    "v": 27
  },
  "result": {
    "type": "BTC",
    "current": {
      "id": "5c49829fdddec563c50f389",
      "type": "BTC",
      "blocknumber": 1453802,
      "timestamp": 1548321429000
    },
    "last": null,
    "namespace": "BTC",
    "sid": "fRrwinMmWUa29PlrAAAB"
  }
}
```

# QUERY AUDIT

## Description

All audit results can be obtained from querying the audit with the audit ID from the step above, including the total deposit value, total withdrawal value and the total internal consumption.

## Example

```
POST localhost:7001/api/v1/audits/5c49829fdddec563c50f389

{
  "deposit_total": "0.1",
  "deposit_num": 1,
  "withdraw_total": "0.08",
  "withdraw_num": 1,
  "sweep_total": "0",
  "sweep_num": 0,
  "sweep_internal_total": "0.01890625",
  "sweep_internal_num": 1,
  "airdrop_total": "0",
  "airdrop_num": 0,
  "recharge_total": "0",
  "recharge_num": 0,
  "recharge_internal_total": "0.01890625",
  "recharge_internal_num": 1,
  "recharge_unknown_total": "0",
  "recharge_unknown_num": 0,
  "recharge_special_total": "0",
  "recharge_special_num": 0,
  "fee_type": "BTC",
  "fee_total": "0.00109375",
  "sweep_fee": "0",
  "sweep_internal_fee": "0",
  "internal_fee": "0",
  "internal_num": 1,
  "extend_fee_total": "0",
  "extend_sweep_fee_total": "0",
  "extend_sweep_internal_fee_total": "0",
  "extend_internal_fee_total": "0",
  "failed_fee_withdraw": "0",
  "failed_withdraw_num": 0,
  "failed_sweep_fee": "0",
  "failed_sweep_num": 0,
  "failed_sweep_internal_fee": "0",
  "failed_sweep_internal_num": 0,
  "failed_fee_internal": "0",
  "failed_internal_num": 0,
  "extend_failed_fee_withdraw": "0",
  "extend_failed_sweep_fee": "0",
  "extend_failed_sweep_internal_fee": "0",
  "extend_failed_fee_internal": "0",
  "type": "BTC",
  "timestamp": 1548321429000,
  "blocknumber": 1453802,
  "create_at": "2019-01-24T09:17:19.035Z",
  "update_at": "2019-01-24T09:17:24.539Z",
  "__v": 0,
  "calc_order_num": 4,
  "id": "5c49829fdddec563c50f389"
}
```