---
title: "Daemon RPC documentation"
---

# Daemon RPC

**`monerod`** [Overview](../interacting/monerod-reference.md)

## Introduction

This is a list of the monerod daemon RPC calls, their inputs and outputs, and examples of each.

Note: "[atomic-units](https://www.getmonero.org/resources/moneropedia/atomic-units.html "Atomic Units refer to the smallest fraction of 1 XMR.")" refer to the smallest fraction of 1 XMR according to the monerod implementation. **1 XMR = 1e12 [atomic-units](https://www.getmonero.org/resources/moneropedia/atomic-units.html "Atomic Units refer to the smallest fraction of 1 XMR.").**

## JSON-RPC example

The API is based on [JSON-RPC standard](https://en.wikipedia.org/wiki/JSON-RPC) version 2.0.

Many `monerod` RPC calls use the daemon's JSON RPC interface while others use their own interfaces, as demonstrated [below](#other-daemon-rpc-methods).

Assuming `monerod` is running on 127.0.0.1:18081, you would call it like this:

```json
IP=127.0.0.1
PORT=18081
METHOD='get_block_header_by_height'
PARAMS='{"height":912345}'
curl \
    http://$IP:$PORT/json_rpc \
    -d '{"jsonrpc":"2.0","id":"0","method":"'$METHOD'","params":'$PARAMS'}' \
    -H 'Content-Type: application/json'
```

If `monerod` was executed with the `--rpc-login` argument as `username:password`, then follow this example:

```json
IP=127.0.0.1
PORT=18081
METHOD='get_block_header_by_height'
PARAMS='{"height":912345}'
curl \
    http://$IP:$PORT/json_rpc \
    -u username:password --digest \
    -d '{"jsonrpc":"2.0","id":"0","method":"'$METHOD'","params":'$PARAMS'}' \
    -H 'Content-Type: application/json'
```

Some methods include parameters, while others do not. Examples of each JSON RPC method follow.

### [JSON RPC Methods](#json-rpc-example):

- [get_block](#get_block)
- [get_block_count](#get_block_count)
- [get_block_header_by_hash](#get_block_header_by_hash)
- [get_block_header_by_height](#get_block_header_by_height)
- [get_block_headers_range](#get_block_headers_range)
- [get_block_template](#get_block_template)
- [get_coinbase_tx_sum](#get_coinbase_tx_sum)
- [get_connections](#get_connections)
- [get_fee_estimate](#get_fee_estimate)
- [get_info](#get_info)
- [get_last_block_header](#get_last_block_header)
- [get_miner_data](#get_miner_data)
- [get_output_distribution](#get_output_distribution)
- [get_output_histogram](#get_output_histogram)
- [get_txpool_backlog](#get_txpool_backlog)
- [get_version](#get_version)
- [hard_fork_info](#hard_fork_info)
- [on_get_block_hash](#on_get_block_hash)
- [prune_blockchain](#prune_blockchain)
- [relay_tx](#relay_tx)
- [set_bans](#set_bans)
- [submit_block](#submit_block)
- [sync_info](#sync_info)

### [Other RPC Methods](#other-daemon-rpc-calls):

- [/get_alt_blocks_hashes](#get_alt_blocks_hashes)
- [/get_blocks.bin](#get_blocksbin)
- [/get_blocks_by_height.bin](#get_blocks_by_heightbin)
- [/get_hashes.bin](#get_hashesbin)
- [/get_height](#get_height)
- [/get_info (not JSON)](#get_info-not-json)
- [/get_limit](#get_limit)
- [/get_net_stats](#get_net_stats)
- [/get_o_indexes.bin](#get_o_indexesbin)
- [/get_output_distribution.bin](#get_output_distributionbin)
- [/get_outs](#get_outs)
- [/get_outs.bin](#get_outsbin)
- [/get_peer_list](#get_peer_list)
- [/get_public_nodes](#get_public_nodes)
- [/get_transaction_pool](#get_transaction_pool)
- [/get_transaction_pool_hashes](#get_transaction_pool_hashes)
- [/get_transaction_pool_hashes.bin](#get_transaction_pool_hashesbin)
- [/get_transaction_pool_stats](#get_transaction_pool_stats)
- [/get_transactions](#get_transactions)
- [/in_peers](#in_peers)
- [/is_key_image_spent](#is_key_image_spent)
- [/mining_status](#mining_status)
- [/out_peers](#out_peers)
- [/pop_blocks](#pop_blocks)
- [/save_bc](#save_bc)
- [/send_raw_transaction](#send_raw_transaction)
- [/set_bootstrap_daemon](#set_bootstrap_daemon)
- [/set_limit](#set_limit)
- [/set_log_categories](#set_log_categories)
- [/set_log_hash_rate](#set_log_hash_rate)
- [/set_log_level](#set_log_level)
- [/start_mining](#start_mining)
- [/start_save_graph](#start_save_graph)
- [/stop_daemon](#stop_daemon)
- [/stop_mining](#stop_mining)
- [/stop_save_graph](#stop_save_graph)
- [/update](#update)

---

### **add_aux_pow**

Easily enable merge mining with Monero without requiring software that manually alters the extra field in the coinbase tx to include the merkle root of the aux blocks.

Alias: _None_ .

Inputs:

- _blocktemplate_blob_ - string;
- _aux_pow_ - array;
  - _id_ - string;
  - _hash_ - string;

Outputs:

- _blocktemplate_blob_ - string;
- _blockhashing_blob_ - string;
- _merkle_root_ - string;
- _merkle_tree_depth_ - unsigned int;
- _aux_pow_ - array;
  - _id_ - string;
  - _hash_ - string;

Example:

```json
$ curl http://127.0.0.1:18081/json_rpc -d
'{"jsonrpc":"2.0","id":"0","method":"add_aux_pow","params":{"blocktemplate_blob":" ...
","aux_pow":
[{"id":"3200b4ea97c3b2081cd4190b58e49572b2319fed00d030ad51809dff06b5d8c8","hash":"7b35762d
e164b20885e15dbe656b1138db06bb402fa1796f5765a23933d8859a"}]}}' -H 'Content-Type:
application/json''

{
  "id": "0",
  "jsonrpc": "2.0",
  "result": {
    "aux_pow": [{
      "hash": "7b35762de164b20885e15dbe656b1138db06bb402fa1796f5765a23933d8859a",
      "id": "3200b4ea97c3b2081cd4190b58e49572b2319fed00d030ad51809dff06b5d8c8"
    }],
    "blockhashing_blob":
"1010ee97e2a106e9f8ebe8887e5b609949ac8ea6143e560ed13552b110cb009b21f0cfca1eaccf00000000b26
85c1283a646bc9020c758daa443be145b7370ce5a6efacb3e614117032e2c22",
    "blocktemplate_blob": " ... ",
    "merkle_root": "7b35762de164b20885e15dbe656b1138db06bb402fa1796f5765a23933d8859a",
    "merkle_tree_depth": 0,
    "status": "OK",
    "untrusted": false
  }
}
```

---

### **banned**

Check if an IP address is banned and for how long.

Alias: _None_

Inputs:

- _address_ - string;

Outputs:

- _status_ - string; General RPC error code. "OK" means everything looks good.
- _banned_ - boolean;

- _seconds_ - unsigned int;

Example:

```json
$ curl http://127.0.0.1:18081/json_rpc -d
'{"jsonrpc":"2.0","id":"0","method":"banned","params":{"address":"95.216.203.255"}}' -H
'Content-Type: application/json'

{
  "id": "0",
  "jsonrpc": "2.0",
  "result": {
    "banned": true,
    "seconds": 690413,
    "status": "OK"
  }
}
```

### **calc_pow**

Calculate PoW hash for a block candidate.

Alias: _None_ .

Inputs:

- _major_version_ - unsigned int; The major version of the monero protocol at this block height.
- _height_ - unsigned int;
- _block_blob_ - blobdata;
- _seed_hash_ - string;

Outputs:

- _result_ - string; PoW hash.

Example:

```json
$ curl http://127.0.0.1:18081/json_rpc -d
'{"jsonrpc":"2.0","id":"0","method":"calc_pow","params":
{"major_version":14,"height":2286447,"block_blob":"0e0ed286da8006ecdc1aab3033cf1716c52f13f
9d8ae0051615a2453643de94643b550d543becd0000000002abc78b0101ffefc68b0101fcfcf0d4b422025014b
b4a1eade6622fd781cb1063381cad396efa69719b41aa28b4fce8c7ad4b5f019ce1dc670456b24a5e03c2d9058
a2df10fec779e2579753b1847b74ee644f16b023c0000000000000000000000000000000000000000000000000
0000000000000000000000000000000000000000000000000000000000000000000000000000000000051399a1bc46a8464
74f5b33db24eae173a26393b976054ee14f9feefe9992523380286709756 4c9db7a36af5bb5ed33ab46e63092b
d8d32cef121608c3258edd55562812e21cc7e3ac73045745a72f7d74581d9a0849d6f30e8b2923171253e864f4
e9ddea3acb5bc755f1c4a878130a70c26297540bc0b7a57affb6b35c1f03d8dbd54ece8457531f8cba15bb7451
6779c01193e212050423020e45aa2c15dcb","seed_hash":"d432f499205150873b2572b5f033c9c6e4b7c6f3
394bd2dd93822cd7085e7307"}}' -H 'Content-Type: application/json'

{
  "id": "0",
  "jsonrpc": "2.0",
  "result": "d0402d6834e26fb94a9ce38c6424d27d2069896a9b8b1ce685d79936bca6e0a8"
}
```

### **flush_cache**

Flush bad transactions / blocks from the cache.

Alias: _None_ .

Inputs:

- _bad_txs_ - boolean; Optional (`false` by default).
- _bad_blocks_ - boolean; Optional (`false` by default).

Outputs:

- _status_ - string; General RPC error code. "OK" means everything looks good.
- _untrusted_ - boolean; States if the result is obtained using the bootstrap mode, and is therefore not trusted (`true`), or when the daemon is fully synced and thus handles the RPC locally (`false`)

Example:

```json
$ curl http://127.0.0.1:18081/json_rpc -d
'{"jsonrpc":"2.0","id":"0","method":"flush_cache","params":
{"bad_txs":true,"bad_blocks":true}}' -H 'Content-Type: application/json'

{
  "id": "0",
  "jsonrpc": "2.0",
  "result": {
    "status": "OK",
    "untrusted": false
  }
}
```

### **flush_txpool**

Flush tx ids from transaction pool

Alias: _None_ .

Inputs:

- _txids_ - array of strings; Optional, list of transactions IDs to flush from pool (all tx ids flushed if empty).

Outputs:

- _status_ - string; General RPC error code. "OK" means everything looks good.

Example:

```json
$ curl http://127.0.0.1:18081/json_rpc -d
'{"jsonrpc":"2.0","id":"0","method":"flush_txpool","params":{"txids":
["dc16fa8eaffe1484ca9014ea050e13131d3acf23b419f33bb4cc0b32b6c49308",""]}}' -H 'Content-
Type: application/json'

{
  "id": "0",
  "jsonrpc": "2.0",
  "result": {
    "status": "OK"
  }
}
```

### **generateblocks**

Generate a block and specify the address to receive the coinbase reward.

Inputs:
- _amount_of_blocks_ - unsigned int; number of blocks to be generated.
- _wallet_address_ - string; address to receive the coinbase reward.
- _prev_block_ - string;
- _starting_nonce_ - unsigned int; Increased by miner until it finds a matching result that solves a block.

Outputs:
- _blocks_ - list of string;
- _height_ - unsigned int;
- _status_ - string; General RPC error code. "OK" means everything looks good.
- _untrusted_ - boolean; States if the result is obtained using the bootstrap mode, and is therefore not trusted (`true`), or when the daemon is fully synced and thus handles the RPC locally (`false`)

The example below uses monerod with the start flags `--regtest --offline --fixed-difficulty 1`. `--offline` ensures that the node does not connect to the main network and learn of its latest chaintip and `--fixed-difficulty` keeps the difficulty constant, allowing a large number of blocks to be generated quickly.

```json
$ curl http://127.0.0.1:18081/json_rpc -d
'{"jsonrpc":"2.0","id":"0","method":"generateblocks","params":
{"amount_of_blocks":1,"wallet_address":"44AFFq5kSiGBoZ4NMDwYtN18obc8AemS33DBLWs3H7otXft3Xj
rpDtQGv7SqSsaBYBb98uNbr2VBBEt7f2wfn3RVGQBEP3A","starting_nonce": 0}' -H 'Content-Type:
application/json'

{
  "id": "0",
  "jsonrpc": "2.0",
  "result": {
    "blocks": ["49b712db7760e3728586f8434ee8bc8d7b3d410dac6bb6e98bf5845c83b917e4"],
    "height": 9783,
    "status": "OK",
    "untrusted": false
  }
}
```

### **get_alternate_chains**

Display alternative chains seen by the node.

Alias: _None_ .

Inputs: _None_ .

Outputs:

- _chains_ - array of chains, the following structure:
  - _block_hash_ - string; the block hash of the first diverging block of this alternative chain.
  - _block_hashes_ - array of strings; An array of all block hashes in the alternative chain that are not in the main chain.
  - _difficulty_ - unsigned int; Least-significant 64 bits of 128-bit integer for the cumulative difficulty of all blocks in the alternative chain.
  - _difficulty_top64_ - unsigned int; Most-significant 64 bits of the 128-bit network difficulty.
  - _height_ - unsigned int; the block height of the last diverging block of this alternative chain.
  - _length_ - unsigned int; the length in blocks of this alternative chain, including the divergent block.
  - _main_chain_parent_block_ - string; The hash of the greatest height block that is shared between the alternative chain and the main chain.
  - _wide_difficulty_ - string; Network difficulty (analogous to the strength of the network) as a hexadecimal string representing a 128-bit number.
- _status_ - string; General RPC error code. "OK" means everything looks good.
- _untrusted_ - boolean; States if the result is obtained using the bootstrap mode, and is

therefore not trusted (`true`), or when the daemon is fully synced and thus handles the RPC locally (`false`)

Example:

```json
$ curl http://127.0.0.1:18081/json_rpc -d
'{"jsonrpc":"2.0","id":"0","method":"get_alternate_chains"}' -H 'Content-Type:
application/json'

{
  "id": "0",
  "jsonrpc": "2.0",
  "result": {
    "chains": [{
      "block_hash": "dd4998cfe92a959a5a0e4ed72432cf23d7dfc4179cbea871ee2a705d71fb5e25",
      "block_hashes":
["dd4998cfe92a959a5a0e4ed72432cf23d7dfc4179cbea871ee2a705d71fb5e25"],
      "difficulty": 86227995333492079,
      "difficulty_top64": 0,
      "height": 2286736,
      "length": 1,
      "main_chain_parent_block":
"6da3d2dc86ccc9353d19fc6b05083125f4ca7d22540d938010462f197a3fe590",
      "wide_difficulty": "0x13257e7a78bfd6f"
    }],
    "status": "OK",
    "untrusted": false
  }
}
```

### **get_bans**

Get list of banned IPs.

Alias: _None_ .

Inputs: _None_ .

Outputs:

- _bans_ - List of banned nodes:
  - _host_ - string; Banned host (IP in A.B.C.D form).
  - _ip_ - unsigned int; Banned IP address, in Int format.
  - _seconds_ - unsigned int; Local Unix time that IP is banned until.
- _status_ - string; General RPC error code. "OK" means everything looks good.
- _untrusted_ - boolean; States if the result is obtained using the bootstrap mode, and is therefore not trusted (`true`), or when the daemon is fully synced and thus handles the RPC locally (`false`)

Example:

```json
$ curl http://127.0.0.1:18081/json_rpc -d '{"jsonrpc":"2.0","id":"0","method":"get_bans"}'
-H 'Content-Type: application/json'

{
  "id": "0",
  "jsonrpc": "2.0",
  "result": {
    "bans": [{
      "host": "102.168.1.51",
      "ip": 855746662,
      "seconds": 22
    },{
      "host": "192.168.1.50",
```

```
      "ip": 838969536,
      "seconds": 28
    }],
    "status": "OK"
    "untrusted": false
  }
}
```

### **get_block**

Full block information can be retrieved by either block height or hash, like with the above block header calls. For full block information, both lookups use the same method, but with different input parameters.

Alias: _getblock_ .

Inputs (pick height or hash):

- _height_ - unsigned int; The block's height.
- _hash_ - string; The block's hash.
- _fill_pow_hash_ - bool; (Optional; Default false) Add PoW hash to block_header response.

Outputs:

- _blob_ - string; Hexadecimal blob of block information.
- _block_header_ - A structure containing block header information. See [get_last_block_header](#get_last_block_header).
- _credits_ - unsigned int; If payment for RPC is enabled, the number of credits available to the requesting client. Otherwise, 0.
- _json_ - json string; JSON formatted block details:
  - _major_version_ - Same as in block header.
  - _minor_version_ - Same as in block header.
  - _timestamp_ - Same as in block header.
  - _prev_id_ - Same as `prev_hash` in block header.
  - _nonce_ - Same as in block header.
  - _miner_tx_ - Miner transaction information
    - _version_ - Transaction version number.
    - _unlock_time_ - The block height when the coinbase transaction becomes spendable.
    - _vin_ - List of transaction inputs:
      - _gen_ - Miner txs are coinbase txs, or "gen".
        - _height_ - This block height, a.k.a. when the coinbase is generated.
    - _vout_ - List of transaction outputs. Each output contains:
      - _amount_ - The amount of the coinbase output, in [atomic-units](https://www.getmonero.org/resources/moneropedia/atomic-units.html "Atomic Units refer to the smallest fraction of 1 XMR.").
      - _target_ -
        - _tagged_key_ -
          - _key_ - the public key of the output
          - _view_tag_ - The 1st byte of a shared secret (used for reducing synchronization time)
    - _extra_ - Usually called the "transaction ID" but can be used to include any random 32 byte/64 character hex string.
    - _rct_signatures_ - Contain signatures of tx signers. Coinbase txs do not have signatures.
  - _tx_hashes_ - List of hashes of non-coinbase transactions in the block. If there are no other transactions, this will be an empty list.
- _status_ - string; General RPC error code. "OK" means everything looks good.
- _top_hash_ - string; If payment for RPC is enabled, the hash of the highest block in the chain. Otherwise, empty.
- _untrusted_ - boolean; States if the result is obtained using the bootstrap mode, and is therefore not trusted (`true`), or when the daemon is fully synced and thus handles the RPC locally (`false`)

**Look up by height:**

In the following example, block 2751506 is looked up by its height. Note that block

2751506 does not have any non-coinbase transactions. (See the next example for a block with extra transactions):

```json
$ curl http://127.0.0.1:18081/json_rpc -d
'{"jsonrpc":"2.0","id":"0","method":"get_block","params":{"height":2751506}}' -H 'Content-
Type: application/json'

{
  "id": "0",
  "jsonrpc": "2.0",
  "result": {
    "blob":
"1010c58bab9b06b27bdecfc6cd0a46172d136c08831cf67660377ba992332363228b1b722781e7807e07f502c
ef8a70101ff92f8a7010180e0a596bb1103d7cbf826b665d7a532c316982dc8dbc24f285cbc18bbcc27c7164cd
9b3277a85d034019f629d8b36bd16a2bfce3ea80c31dc4d8762c67165aec21845494e32b7582fe002110000002
97a787a00000000000000000000000",
    "block_header": {
      "block_size": 106,
      "block_weight": 106,
      "cumulative_difficulty": 236046001376524168,
      "cumulative_difficulty_top64": 0,
      "depth": 40,
      "difficulty": 313732272488,
      "difficulty_top64": 0,
      "hash": "43bd1f2b6556dcafa413d8372974af59e4e8f37dbf74dc6b2a9b7212d0577428",
      "height": 2751506,
      "long_term_weight": 176470,
      "major_version": 16,
      "miner_tx_hash": "e49b854c5f339d7410a77f2a137281d8042a0ffc7ef9ab24cd670b67139b24cd",
      "minor_version": 16,
      "nonce": 4110909056,
      "num_txes": 0,
      "orphan_status": false,
      "pow_hash": "",
      "prev_hash": "b27bdecfc6cd0a46172d136c08831cf67660377ba992332363228b1b722781e7",
      "reward": 600000000000,
      "timestamp": 1667941829,
      "wide_cumulative_difficulty": "0x3469a966eb2f788",
      "wide_difficulty": "0x490be69168"
    },
    "credits": 0,
    "json": "{\n  \"major_version\": 16, \n  \"minor_version\": 16, \n  \"timestamp\":
1667941829, \n  \"prev_id\":
\"b27bdecfc6cd0a46172d136c08831cf67660377ba992332363228b1b722781e7\", \n  \"nonce\":
4110909056, \n  \"miner_tx\": {\n    \"version\": 2, \n    \"unlock_time\": 2751566, \n
\"vin\": [ {\n        \"gen\": {\n          \"height\": 2751506\n        }\n      }\n
], \n    \"vout\": [ {\n        \"amount\": 600000000000, \n        \"target\": {\n
\"tagged_key\": {\n            \"key\":
\"d7cbf826b665d7a532c316982dc8dbc24f285cbc18bbcc27c7164cd9b3277a85\", \n
\"view_tag\": \"d0\"\n          }\n        }\n      }\n    ], \n    \"extra\": [ 1, 159,
98, 157, 139, 54, 189, 22, 162, 191, 206, 62, 168, 12, 49, 220, 77, 135, 98, 198, 113,
101, 174, 194, 24, 69, 73, 78, 50, 183, 88, 47, 224, 2, 17, 0, 0, 0, 41, 122, 120, 122, 0,
0, 0, 0, 0, 0, 0, 0, 0, 0\n    ], \n    \"rct_signatures\": {\n      \"type\": 0\n    }\n
}, \n  \"tx_hashes\": [ ]\n}",
    "miner_tx_hash": "e49b854c5f339d7410a77f2a137281d8042a0ffc7ef9ab24cd670b67139b24cd",
    "status": "OK",
    "top_hash": "",
    "untrusted": false
  }
}
```

**Look up by hash:**

In the following example, block 2751210 is looked up by its hash. Note that block 2751210 has 2 non-coinbase transactions:

```json
```

```
$ curl http://127.0.0.1:18081/json_rpc -d
'{"jsonrpc":"2.0","id":"0","method":"get_block","params":
{"hash":"86d421322b700166dde2d7eba1cc8600925ef640abf6c0a2cc8ce0d6dd90abfd"}' -H 'Content-
Type: application/json'

{
  "id": "0",
  "jsonrpc": "2.0",
  "result": {
    "blob":
"1010d8faa89b06f8a36d0dbe4d27d2f52160000563896048d71067c31e99a3869bf9b7142227bb5328010b02a
6f6a70101ffeaf5a70101a08bc8b3bb11036d6713f5aa552a1aaf33baed7591f795b86daf339e51029a9062dfe
09f0f909b312b0124d6023d591c4d434000e5e31c6db718a1e96e865939930e90a7042a1cd4cbd202083786a78
452fdfc000002a89e380a44d8dfc64b551baa171447a0f9c9262255be6e8f8ef10896e36e2bf90c4d343e416e3
94ad9cc10b7d2df7b2f39370a554730f75dfcb04944bd62c299",
    "block_header": {
      "block_size": 3166,
      "block_weight": 3166,
      "cumulative_difficulty": 235954020187853162,
      "cumulative_difficulty_top64": 0,
      "depth": 26,
      "difficulty": 312527777859,
      "difficulty_top64": 0,
      "hash": "86d421322b700166dde2d7eba1cc8600925ef640abf6c0a2cc8ce0d6dd90abfd",
      "height": 2751210,
      "long_term_weight": 176470,
      "major_version": 16,
      "miner_tx_hash": "dabe07900d3123ed895612f4a151adb3e39681b145f0f85bfee23ea1fe47acf2",
      "minor_version": 16,
      "nonce": 184625235,
      "num_txes": 2,
      "orphan_status": false,
      "pow_hash": "",
      "prev_hash": "f8a36d0dbe4d27d2f52160000563896048d71067c31e99a3869bf9b7142227bb",
      "reward": 600061380000,
      "timestamp": 1667906904,
      "wide_cumulative_difficulty": "0x34646ee649f516a",
      "wide_difficulty": "0x48c41b7043"
    },
    "credits": 0,
    "json": "{\n  \"major_version\": 16, \n  \"minor_version\": 16, \n  \"timestamp\":
1667906904, \n  \"prev_id\":
\"f8a36d0dbe4d27d2f52160000563896048d71067c31e99a3869bf9b7142227bb\", \n  \"nonce\":
184625235, \n  \"miner_tx\": {\n    \"version\": 2, \n    \"unlock_time\": 2751270, \n
\"vin\": [ {\n        \"gen\": {\n          \"height\": 2751210\n        }\n      }\n
], \n    \"vout\": [ {\n        \"amount\": 600061380000, \n        \"target\": {\n
\"tagged_key\": {\n            \"key\":
\"6d6713f5aa552a1aaf33baed7591f795b86daf339e51029a9062dfe09f0f909b\", \n
\"view_tag\": \"31\"\n          }\n        }\n      }\n    ], \n    \"extra\": [ 1, 36,
214, 2, 61, 89, 28, 77, 67, 64, 0, 229, 227, 28, 109, 183, 24, 161, 233, 110, 134, 89, 57,
147, 14, 144, 167, 4, 42, 28, 212, 203, 210, 2, 8, 55, 134, 167, 132, 82, 253, 252, 0\n
], \n    \"rct_signatures\": {\n      \"type\": 0\n    }\n  }, \n  \"tx_hashes\": [
\"a89e380a44d8dfc64b551baa171447a0f9c9262255be6e8f8ef10896e36e2bf9\",
\"0c4d343e416e394ad9cc10b7d2df7b2f39370a554730f75dfcb04944bd62c299\"\n  ]\n}",
    "miner_tx_hash": "dabe07900d3123ed895612f4a151adb3e39681b145f0f85bfee23ea1fe47acf2",
    "status": "OK",
    "top_hash": "",
    "tx_hashes":
["a89e380a44d8dfc64b551baa171447a0f9c9262255be6e8f8ef10896e36e2bf9","0c4d343e416e394ad9cc1
0b7d2df7b2f39370a554730f75dfcb04944bd62c299"],
    "untrusted": false
  }
}
```

### **get_block_count**

Look up how many blocks are in the longest chain known to the node.

Alias: _getblockcount_ .

Inputs: _None_ .

Outputs:

- _count_ - unsigned int; Number of blocks in longest chain seen by the node.
- _status_ - string; General RPC error code. "OK" means everything looks good.
- _untrusted_ - boolean; States if the result is obtained using the bootstrap mode, and is therefore not trusted (`true`), or when the daemon is fully synced and thus handles the RPC locally (`false`)

Example:

```json
$ curl http://127.0.0.1:18081/json_rpc -d
'{"jsonrpc":"2.0","id":"0","method":"get_block_count"}' -H 'Content-Type:
application/json'

{
  "id": "0",
  "jsonrpc": "2.0",
  "result": {
    "count": 993163,
    "status": "OK"
    "untrusted": "false"
  }
}
```

### **get_block_header_by_hash**

Block header information can be retrieved using either a block's hash or height. This method includes a block's hash as an input parameter to retrieve basic information about the block.

Alias: _getblockheaderbyhash_ .

Inputs:

- _hash_ - string; The sha256 hash of the block.
- _hashes_ - array of strings; The sha256 hashes of the blocks.
- _fill_pow_hash_ - boolean; (Optional; defaults to `false`) Add PoW hash to block_header response.

Outputs:

- _block_header_ - A structure containing block header information. See [get_last_block_header](#get_last_block_header).
- _status_ - string; General RPC error code. "OK" means everything looks good.
- _untrusted_ - boolean; States if the result is obtained using the bootstrap mode, and is therefore not trusted (`true`), or when the daemon is fully synced and thus handles the RPC locally (`false`)

In this example, block 912345 is looked up by its hash:

```json
$ curl http://127.0.0.1:18081/json_rpc -d
'{"jsonrpc":"2.0","id":"0","method":"get_block_header_by_hash","params":
{"hash":"e22cf75f39ae720e8b71b3d120a5ac03f0db50bba6379e2850975b4859190bc6"}}' -H 'Content-
Type: application/json'

{
  "id": "0",
  "jsonrpc": "2.0",
```

```
  "result": {
    "block_header": {
      "block_size": 210,
      "block_weight": 210,
      "cumulative_difficulty": 754734824984346,
      "cumulative_difficulty_top64": 0,
      "depth": 1374113,
      "difficulty": 815625611,
      "difficulty_top64": 0,
      "hash": "e22cf75f39ae720e8b71b3d120a5ac03f0db50bba6379e2850975b4859190bc6",
      "height": 912345,
      "long_term_weight": 210,
      "major_version": 1,
      "miner_tx_hash": "c7da3965f25c19b8eb7dd8db48dcd4e7c885e2491db77e289f0609bf8e08ec30",
      "minor_version": 2,
      "nonce": 1646,
      "num_txes": 0,
      "orphan_status": false,
      "pow_hash": "",
      "prev_hash": "b61c58b2e0be53fad5ef9d9731a55e8a81d972b8d90ed07c04fd37ca6403ff78",
      "reward": 7388968946286,
      "timestamp": 1452793716,
      "wide_cumulative_difficulty": "0x2ae6d65248f1a",
      "wide_difficulty": "0x309d758b"
    },
    "credits": 0,
    "status": "OK",
    "top_hash": "",
    "untrusted": false
  }
}
```


### **get_block_header_by_height**

Similar to [get_block_header_by_hash](#get_block_header_by_hash) above, this method includes a block's height as an input parameter to retrieve basic information about the block.

Alias: _getblockheaderbyheight_ .

Inputs:

- _height_ - unsigned int; The block's height.
- _fill_pow_hash_ - boolean; (Optional; defaults to `false`) Add PoW hash to block_header response.

Outputs:

- _block_header_ - A structure containing block header information. See [get_last_block_header](#get_last_block_header).
- _status_ - string; General RPC error code. "OK" means everything looks good.
- _untrusted_ - boolean; States if the result is obtained using the bootstrap mode, and is therefore not trusted (`true`), or when the daemon is fully synced and thus handles the RPC locally (`false`)

In this example, block 912345 is looked up by its height (notice that the returned information is the same as in the previous example):

```json
$ curl http://127.0.0.1:18081/json_rpc -d
'{"jsonrpc":"2.0","id":"0","method":"get_block_header_by_height","params":
{"height":912345}}' -H 'Content-Type: application/json'

{
  "id": "0",
  "jsonrpc": "2.0",
```

```
  "result": {
    "block_header": {
      "block_size": 210,
      "block_weight": 210,
      "cumulative_difficulty": 754734824984346,
      "cumulative_difficulty_top64": 0,
      "depth": 1374118,
      "difficulty": 815625611,
      "difficulty_top64": 0,
      "hash": "e22cf75f39ae720e8b71b3d120a5ac03f0db50bba6379e2850975b4859190bc6",
      "height": 912345,
      "long_term_weight": 210,
      "major_version": 1,
      "miner_tx_hash": "c7da3965f25c19b8eb7dd8db48dcd4e7c885e2491db77e289f0609bf8e08ec30",
      "minor_version": 2,
      "nonce": 1646,
      "num_txes": 0,
      "orphan_status": false,
      "pow_hash": "",
      "prev_hash": "b61c58b2e0be53fad5ef9d9731a55e8a81d972b8d90ed07c04fd37ca6403ff78",
      "reward": 7388968946286,
      "timestamp": 1452793716,
      "wide_cumulative_difficulty": "0x2ae6d65248f1a",
      "wide_difficulty": "0x309d758b"
    },
    "credits": 0,
    "status": "OK",
    "top_hash": "",
    "untrusted": false
  }
}
```


### **get_block_headers_range**

Similar to [get_block_header_by_height](#get_block_header_by_height) above, but for a range of blocks. This method includes a starting block height and an ending block height as parameters to retrieve basic information about the range of blocks.

Alias: _getblockheadersrange_ .

Inputs:

- _start_height_ - unsigned int; The starting block's height.
- _end_height_ - unsigned int; The ending block's height.
- _fill_pow_hash_ - boolean; (Optional; defaults to `false`) Add PoW hash to block_header response.

Outputs:

- _credits_ - unsigned int; If payment for RPC is enabled, the number of credits available to the requesting client. Otherwise, 0.
- _headers_ - array of `block_header` (a structure containing block header information. See [get_last_block_header](#get_last_block_header)).
- _status_ - string; General RPC error code. "OK" means everything looks good.
- _top_hash_ - string; If payment for RPC is enabled, the hash of the highest block in the chain. Otherwise, empty.
- _untrusted_ - boolean; States if the result is obtained using the bootstrap mode, and is therefore not trusted (`true`), or when the daemon is fully synced and thus handles the RPC locally (`false`)

In this example, blocks range from height 1545999 to 1546000 is looked up (notice that the returned information are ascending order and that it is at the April 2018 network upgrade time):

```json
$ curl http://127.0.0.1:18081/json_rpc -d
'{"jsonrpc":"2.0","id":"0","method":"get_block_headers_range","params":
```

```
{"start_height":1545999,"end_height":1546000}}' -H 'Content-Type: application/json'

{
  "id": "0",
  "jsonrpc": "2.0",
  "result": {
    "credits": 0,
    "headers": [{
      "block_size": 301413,
      "block_weight": 301413,
      "cumulative_difficulty": 13185267971483472,
      "cumulative_difficulty_top64": 0,
      "depth": 740464,
      "difficulty": 134636057921,
      "difficulty_top64": 0,
      "hash": "86d1d20a40cefcf3dd410ff6967e0491613b77bf73ea8f1bf2e335cf9cf7d57a",
      "height": 1545999,
      "long_term_weight": 301413,
      "major_version": 6,
      "miner_tx_hash": "9909c6f8a5267f043c3b2b079fb4eacc49ef9c1dee1c028eeb1a259b95e6e1d9",
      "minor_version": 6,
      "nonce": 3246403956,
      "num_txes": 20,
      "orphan_status": false,
      "pow_hash": "",
      "prev_hash": "0ef6e948f77b8f8806621003f5de24b1bcbea150bc0e376835aea099674a5db5",
      "reward": 5025593029981,
      "timestamp": 1523002893,
      "wide_cumulative_difficulty": "0x2ed7ee6db56750",
      "wide_difficulty": "0x1f58ef3541"
    },{
      "block_size": 13322,
      "block_weight": 13322,
      "cumulative_difficulty": 13185402687569710,
      "cumulative_difficulty_top64": 0,
      "depth": 740463,
      "difficulty": 134716086238,
      "difficulty_top64": 0,
      "hash": "b408bf4cfcd7de13e7e370c84b8314c85b24f0ba4093ca1d6eeb30b35e34e91a",
      "height": 1546000,
      "long_term_weight": 13322,
      "major_version": 7,
      "miner_tx_hash": "7f749c7c64acb35ef427c7454c45e6688781fbead9bbf222cb12ad1a96a4e8f6",
      "minor_version": 7,
      "nonce": 3737164176,
      "num_txes": 1,
      "orphan_status": false,
      "pow_hash": "",
      "prev_hash": "86d1d20a40cefcf3dd410ff6967e0491613b77bf73ea8f1bf2e335cf9cf7d57a",
      "reward": 4851952181070,
      "timestamp": 1523002931,
      "wide_cumulative_difficulty": "0x2ed80dcb69bf2e",
      "wide_difficulty": "0x1f5db457de"
    }],
    "status": "OK",
    "top_hash": "",
    "untrusted": false
  }
}
```

### **get_block_template**

Get a block template on which mining a new block.

Alias: _getblocktemplate_ .

Inputs:

- _wallet_address_ - string; Address of wallet to receive coinbase transaction if block is successfully mined.
- _reserve_size_ - unsigned int; Reserve size.

Outputs:

- _blocktemplate_blob_ - string; Blob on which to try to mine a new block.
- _blockhashing_blob_ - string; Blob on which to try to find a valid nonce.
- _difficulty_ - unsigned int; Least-significant 64 bits of the 128-bit network difficulty.
- _difficulty_top64_ - unsigned int; Most-significant 64 bits of the 128-bit network difficulty.
- _expected_reward_ - unsigned int; Coinbase reward expected to be received if block is successfully mined.
- _height_ - unsigned int; Height on which to mine.
- _next_seed_hash_ - string; Hash of the next block to use as seed for Random-X proof-of-work.
- _prev_hash_ - string; Hash of the most recent block on which to mine the next block.
- _reserved_offset_ - unsigned int; Reserved offset.
- _seed_hash_ - string; Hash of block to use as seed for Random-X proof-of-work.
- _seed_height_ - unsigned int; Height of block to use as seed for Random-X proof-of-work.
- _status_ - string; General RPC error code. "OK" means everything looks good.
- _untrusted_ - boolean; States if the result is obtained using the bootstrap mode, and is therefore not trusted (`true`), or when the daemon is fully synced and thus handles the RPC locally (`false`)
- _wide_difficulty_ - string; Network difficulty (analogous to the strength of the network) as a hexadecimal string representing a 128-bit number.

Example:

```json
$ curl http://127.0.0.1:18081/json_rpc -d
'{"jsonrpc":"2.0","id":"0","method":"get_block_template","params":
{"wallet_address":"44GBHzv6ZyQdJkjqZje6KLZ3xSyN1hBSFAnLP6EAqJtCRVzMzZmeXTC2AHKDS9aEDTRKmo6
a6o9r9j86pYfhCWDkKjbtcns","reserve_size":60}}' -H 'Content-Type: application/json'

{
  "id": "0",
  "jsonrpc": "2.0",
  "result": {
    "blockhashing_blob":
"0e0ed286da8006ecdc1aab3033cf1716c52f13f9d8ae0051615a2453643de94643b550d543becd00000000d13
0d22cf308b308498bbc16e2e955e7dbd691e6a8fab805f98ad82e6faa8bcc06",
    "blocktemplate_blob":
"0e0ed286da8006ecdc1aab3033cf1716c52f13f9d8ae0051615a2453643de94643b550d543becd0000000002a
bc78b0101ffefc68b0101fcfcf0d4b422025014bb4a1eade6622fd781cb1063381cad396efa69719b41aa28b4f
ce8c7ad4b5f019ce1dc670456b24a5e03c2d9058a2df10fec779e2579753b1847b74ee644f16b023c000000000
0000000000000000000000000000000000000000000000000000000000000000000000000000000000000000
0000000000000000000000051399a1bc46a846474f5b33db24eae173a26393b976054ee14f9feefe999252338
02867097564c9db7a36af5bb5ed33ab46e63092bd8d32cef121608c3258edd55562812e21cc7e3ac73045745a7
2f7d74581d9a0849d6f30e8b2923171253e864f4e9ddea3acb5bc755f1c4a878130a70c26297540bc0b7a57aff
b6b35c1f03d8dbd54ece8457531f8cba15bb74516779c01193e212050423020e45aa2c15dcb",
    "difficulty": 226807339040,
    "difficulty_top64": 0,
    "expected_reward": 1182367759996,
    "height": 2286447,
    "next_seed_hash": "",
    "prev_hash": "ecdc1aab3033cf1716c52f13f9d8ae0051615a2453643de94643b550d543becd",
    "reserved_offset": 130,
    "seed_hash": "d432f499205150873b2572b5f033c9c6e4b7c6f3394bd2dd93822cd7085e7307",
    "seed_height": 2285568,
    "status": "OK",
    "untrusted": false,
    "wide_difficulty": "0x34cec55820"
  }
```

### **get_coinbase_tx_sum**

Get the newly emitted coins amount and the fees amount for n last blocks starting at particular height

Alias: _None_ .

Inputs:

- _height_ - unsigned int; Block height from which getting the amounts
- _count_ - unsigned int; number of blocks to include in the sum

Outputs:

- _credits_ - unsigned int; If payment for RPC is enabled, the number of credits available to the requesting client. Otherwise, 0.
- _emission_amount_ - unsigned int; Least significant 64 bits for 128 bit integer representing the new coins emitted in [atomic-units](https://www.getmonero.org/resources/moneropedia/atomic-units.html "Atomic Units refer to the smallest fraction of 1 XMR."). (See src/rpc/core_rpc_server.cpp store_128)
- _emission_amount_top64_ - unsigned it; Most significant 64 bits for 128 bit integer representing the new coins emitted in [atomic-units](https://www.getmonero.org/resources/moneropedia/atomic-units.html "Atomic Units refer to the smallest fraction of 1 XMR.")
- _fee_amount_ - unsigned int; Most significant 64 bits for 128 bit integer representing the sum of fees in [atomic-units](https://www.getmonero.org/resources/moneropedia/atomic-units.html "Atomic Units refer to the smallest fraction of 1 XMR.").
- _fee_amount_top64_ - unsigned int; Most significant 64 bits for 128 bit integer representing the sum of fees in [atomic-units](https://www.getmonero.org/resources/moneropedia/atomic-units.html "Atomic Units refer to the smallest fraction of 1 XMR.").
- _status_ - string; General RPC error code. "OK" means everything looks good.
- _top_hash_ - string; If payment for RPC is enabled, the hash of the highest block in the chain. Otherwise, empty.
- _untrusted_ - boolean; States if the result is obtained using the bootstrap mode, and is therefore not trusted (`true`), or when the daemon is fully synced and thus handles the RPC locally (`false`)
- _wide_emission_amount_ - string (128 bit hex encoded integer); New coins emitted in [atomic-units](https://www.getmonero.org/resources/moneropedia/atomic-units.html "Atomic Units refer to the smallest fraction of 1 XMR.").
- _wide_fee_amount_ - string (128 bit hex encoded integer); Sum of fees in [atomic-units](https://www.getmonero.org/resources/moneropedia/atomic-units.html "Atomic Units refer to the smallest fraction of 1 XMR.").

Example:

```json
$ curl http://127.0.0.1:18081/json_rpc -d
'{"jsonrpc":"2.0","id":"0","method":"get_coinbase_tx_sum","params":
{"height":1563078,"count":2}}' -H 'Content-Type: application/json'

{
  "id": "0",
  "jsonrpc": "2.0",
  "result": {
    "credits": 0,
    "emission_amount": 9471836197320,
    "emission_amount_top64": 0,
    "fee_amount": 0,
    "fee_amount_top64": 0,
    "status": "OK",
    "top_hash": "",
    "untrusted": false,
    "wide_emission_amount": "0x89d556e91c8",
    "wide_fee_amount": "0x0"
  }
}
```

```

### **get_connections**

Retrieve information about incoming and outgoing connections to your node.

Alias: _None_ .

Inputs: _None_ .

Outputs:

- _connections_ - List of all connections and their info:
  - _address_ - string; The peer's address, actually IPv4 & port.
  - _address_type_ - unsigned int; 0 = invalid, 1 = ipv4 and 2 = ipv6.
  - _avg_download_ - unsigned int; Average bytes of data downloaded by node.
  - _avg_upload_ - unsigned int; Average bytes of data uploaded by node.
  - _connection_id_ - string; The connection ID
  - _current_download_ - unsigned int; Current bytes downloaded by node.
  - _current_upload_ - unsigned int; Current bytes uploaded by node.
  - _height_ - unsigned int; The peer height
  - _host_ - string; The peer host
  - _incoming_ - boolean; Is the node getting information from your node?
  - _ip_ - string; The node's IP address.
  - _live_time_ - unsigned int
  - _local_ip_ - boolean
  - _localhost_ - boolean
  - _peer_id_ - string; The node's ID on the network.
  - _port_ - string; The port that the node is using to connect to the network.
  - _recv_count_ - unsigned int
  - _recv_idle_time_ - unsigned int
  - _rpc_credits_per_hash_ - unsigned int; clients will be awarded credits/difficulty credits for every hash they calculate.
  - _rpc_port_ - unsigned int
  - _send_count_ - unsigned int
  - _send_idle_time_ - unsigned int
  - _state_ - string
  - _support_flags_ - unsigned int
- _status_ - string; General RPC error code. "OK" means everything looks good.
- _$1_ - boolean; States if the result is obtained using the bootstrap mode, and is therefore not trusted (`true`), or when the daemon is fully synced and thus handles the RPC locally (`false`)

Following is an example of `get_connections` and it's return:

```json
$ curl http://127.0.0.1:18081/json_rpc -d
'{"jsonrpc":"2.0","id":"0","method":"get_connections"}' -H 'Content-Type:
application/json'

{
  "id": "0",
  "jsonrpc": "2.0",
  "result": {
    "connections": [{
      "address": "51.75.162.171:44741",
      "address_type": 1,
      "avg_download": 0,
      "avg_upload": 1,
      "connection_id": "4420a6fcf9c642daaae41400ccfc1fd7",
      "current_download": 0,
      "current_upload": 1,
      "height": 1,
      "host": "51.75.162.171",
      "incoming": true,
      "ip": "51.75.162.171",
      "live_time": 9,
```
```

```
      "local_ip": false,
      "localhost": false,
      "peer_id": "ff561b6a65c2838c",
      "port": "44741",
      "pruning_seed": 0,
      "recv_count": 382,
      "recv_idle_time": 8,
      "rpc_credits_per_hash": 0,
      "rpc_port": 0,
      "send_count": 15434,
      "send_idle_time": 8,
      "state": "normal",
      "support_flags": 1
    },{

    ...
    }],
    "status": "OK"
    "untrusted": false
  }
}
```

### **get_fee_estimate**

Gives an estimation on fees per byte.

Alias: _None_ .

Inputs:

- _grace_blocks_ - unsigned int; Optional

Outputs:

- _credits_ - unsigned int; If payment for RPC is enabled, the number of credits available to the requesting client. Otherwise, 0.
- _fee_ - unsigned int; Amount of fees estimated per byte in [atomic-units](https://www.getmonero.org/resources/moneropedia/atomic-units.html "Atomic Units refer to the smallest fraction of 1 XMR.")
- _fees_ - array of unsigned int; Represents the base fees at different priorities [slow, normal, fast, fastest].
- _quantization_mask_ - unsigned int; Final fee should be rounded up to an even multiple of this value
- _status_ - string; General RPC error code. "OK" means everything looks good.
- _top_hash_ - string; If payment for RPC is enabled, the hash of the highest block in the chain. Otherwise, empty.
- _untrusted_ - boolean; States if the result is obtained using the bootstrap mode, and is therefore not trusted (`true`), or when the daemon is fully synced and thus handles the RPC locally (`false`)

Example:

```json
$ curl http://127.0.0.1:18081/json_rpc -d
'{"jsonrpc":"2.0","id":"0","method":"get_fee_estimate"}' -H 'Content-Type:
application/json'

{
  "id": "0",
  "jsonrpc": "2.0",
  "result": {
    "credits": 0,
    "fee": 7874,
    "fees": [20000,80000,320000,4000000],
    "quantization_mask": 10000,
    "status": "OK",
```

```
    "top_hash": "",
    "untrusted": false
  }
}
```

### **get_info**

Retrieve general information about the state of your node and the network.

Alias:

- _/get_info_
- _/getinfo_

See other RPC Methods [/get_info (not JSON)](#get_info-not-json)

Inputs: _None_ .

Outputs:

- _adjusted_time_ - unsigned int; Current time approximated from chain data, as Unix time.
- _alt_blocks_count_ - unsigned int; Number of alternative blocks to main chain.
- _block_size_limit_ - unsigned int; Backward compatibility, same as _block_weight_limit_ , use that instead
- _block_size_median_ - unsigned int; Backward compatibility, same as _block_weight_median_ , use that instead
- _block_weight_limit_ - unsigned int; Maximum allowed adjusted block size based on latest 100000 blocks
- _block_weight_median_ - unsigned int; Median adjusted block size of latest 100000 blocks
- _bootstrap_daemon_address_ - string; @Bootstrap-node to give immediate usability to wallets while syncing by proxying RPC to it. (Note: the replies may be untrustworthy).
- _busy_syncing_ - boolean; States if new blocks are being added (`true`) or not (`false`).
- _credits_ - unsigned int; If payment for RPC is enabled, the number of credits available to the requesting client. Otherwise, 0.
- _cumulative_difficulty_ - unsigned int; Least-significant 64 bits of the 128-bit cumulative difficulty.
- _cumulative_difficulty_top64_ - unsigned int; Most-significant 64 bits of the 128-bit cumulative difficulty.
- _database_size_ - unsigned int; The size of the blockchain database, in bytes.
- _difficulty_ - unsigned int; Least-significant 64 bits of the 128-bit network difficulty.
- _difficulty_top64_ - unsigned int; Most-significant 64 bits of the 128-bit network difficulty.
- _free_space_ - unsigned int; Available disk space on the node.
- _grey_peerlist_size_ - unsigned int; Grey Peerlist Size
- _height_ - unsigned int; Current length of longest chain known to daemon.
- _height_without_bootstrap_ - unsigned int; Current length of the local chain of the daemon.
- _incoming_connections_count_ - unsigned int; Number of peers connected to and pulling from your node.
- _mainnet_ - boolean; States if the node is on the mainnet (`true`) or not (`false`).
- _nettype_ - string; Network type (one of `mainnet`, `stagenet` or `testnet`).
- _offline_ - boolean; States if the node is offline (`true`) or online (`false`).
- _outgoing_connections_count_ - unsigned int; Number of peers that you are connected to and getting information from.
- _rpc_connections_count_ - unsigned int; Number of RPC client connected to the daemon (Including this RPC request).
- _stagenet_ - boolean; States if the node is on the stagenet (`true`) or not (`false`).
- _start_time_ - unsigned int; Start time of the daemon, as UNIX time.
- _status_ - string; General RPC error code. "OK" means everything looks good.
- _synchronized_ - boolean; States if the node is synchronized (`true`) or not (`false`).
- _target_ - unsigned int; Current target for next proof of work.
- _target_height_ - unsigned int; The height of the next block in the chain.
- _testnet_ - boolean; States if the node is on the testnet (`true`) or not (`false`).
- _top_block_hash_ - string; Hash of the highest block in the chain.

- _top_hash_ - string; If payment for RPC is enabled, the hash of the highest block in the chain. Otherwise, empty.
- _tx_count_ - unsigned int; Total number of non-coinbase transactions in the chain.
- _tx_pool_size_ - unsigned int; Number of transactions that have been broadcast but not included in a block.
- _$1_ - boolean; States if the result is obtained using the bootstrap mode, and is therefore not trusted (`true`), or when the daemon is fully synced and thus handles the RPC locally (`false`)
- _update_available_ - boolean; States if a newer Monero software version is available.
- _version_ - string; The version of the Monero software the node is running.
- _was_bootstrap_ever_used_ - boolean; States if a bootstrap node has ever been used since the daemon started.
- _white_peerlist_size_ - unsigned int; White Peerlist Size
- _wide_cumulative_difficulty_ - Cumulative difficulty of all blocks in the blockchain as a hexadecimal string representing a 128-bit number.
- _wide_difficulty_ - string; Network difficulty (analogous to the strength of the network) as a hexadecimal string representing a 128-bit number.

Following is an example `get_info` call and its return:

```json
$ curl http://127.0.0.1:18081/json_rpc -d '{"jsonrpc":"2.0","id":"0","method":"get_info"}'
-H 'Content-Type: application/json'

{
  "id": "0",
  "jsonrpc": "2.0",
  "result": {
    "adjusted_time": 1612090533,
    "alt_blocks_count": 2,
    "block_size_limit": 600000,
    "block_size_median": 300000,
    "block_weight_limit": 600000,
    "block_weight_median": 300000,
    "bootstrap_daemon_address": "",
    "busy_syncing": false,
    "credits": 0,
    "cumulative_difficulty": 86168732847545368,
    "cumulative_difficulty_top64": 0,
    "database_size": 34329849856,
    "difficulty": 225889137349,
    "difficulty_top64": 0,
    "free_space": 10795802624,
    "grey_peerlist_size": 4999,
    "height": 2286472,
    "height_without_bootstrap": 2286472,
    "incoming_connections_count": 85,
    "mainnet": true,
    "nettype": "mainnet",
    "offline": false,
    "outgoing_connections_count": 16,
    "rpc_connections_count": 1,
    "stagenet": false,
    "start_time": 1611915662,
    "status": "OK",
    "synchronized": true,
    "target": 120,
    "target_height": 2286464,
    "testnet": false,
    "top_block_hash": "b92720d8315b96e32020d04e14a0c54cc13e057d4a5beb4501be490d306fdd8f",
    "top_hash": "",
    "tx_count": 11239803,
    "tx_pool_size": 21,
    "untrusted": false,
    "update_available": false,
    "version": "0.17.1.9-release",
    "was_bootstrap_ever_used": false,
    "white_peerlist_size": 1000,
    "wide_cumulative_difficulty": "0x1322201881f9c18",
```

```
    "wide_difficulty": "0x34980ab2c5"
  }
}
```

### **get_last_block_header**

Block header information for the most recent block is easily retrieved with this method. No inputs are needed.

Alias: _getlastblockheader_ .

Inputs:

- _fill_pow_hash_ - boolean; (Optional; defaults to `false`) Add PoW hash to block_header response.

Outputs:

- _block_header_ - A structure containing block header information.
  - _block_size_ - unsigned int; Backward compatibility, same as _block_weight_ , use that instead
  - _block_weight_ - unsigned int; The adjusted block size, in bytes. This is the raw size, plus a positive adjustment for any Bulletproof transactions with more than 2 outputs.
  - _cumulative_difficulty_ - unsigned int; Least-significant 64 bits of the cumulative difficulty of all blocks up to the block in the reply.
  - _cumulative_difficulty_top64_ - unsigned int; Most-significant 64 bits of the 128-bit cumulative difficulty.
  - _depth_ -  unsigned int; The number of blocks succeeding this block on the blockchain. A larger number means an older block.
  - _difficulty_ - unsigned int; The strength of the Monero network based on mining power.
  - _difficulty_top64_ - unsigned int; Most-significant 64 bits of the 128-bit network difficulty.
  - _hash_ - string; The hash of this block.
  - _height_ - unsigned int; The number of blocks preceding this block on the blockchain.
  - _long_term_weight_ - unsigned int; The long term block weight, based on the median weight of the preceding 100000 blocks.
  - _major_version_ - unsigned int; The major version of the monero protocol at this block height.
  - _miner_tx_hash_ - string; The hash of this block's coinbase transaction.
  - _minor_version_ - unsigned int; The minor version of the monero protocol at this block height.
  - _nonce_ - unsigned int; a cryptographic random one-time number used in mining a Monero block.
  - _num_txes_ - unsigned int; Number of transactions in the block, not counting the coinbase tx.
  - _orphan_status_ - boolean; Usually `false`. If `true`, this block is not part of the longest chain.
  - _pow_hash_ - string; The hash, as a hexadecimal string, calculated from the block as proof-of-work.
  - _prev_hash_ - string; The hash of the block immediately preceding this block in the chain.
  - _reward_ - unsigned int; The amount of [atomic-units](https://www.getmonero.org/resources/moneropedia/atomic-units.html "Atomic Units refer to the smallest fraction of 1 XMR.") rewarded to the miner. The reward is the sum of new coins created (the emission) and fees paid by transactions in this block. Note: 1 XMR = 1e12 [atomic-units](https://www.getmonero.org/resources/moneropedia/atomic-units.html "Atomic Units refer to the smallest fraction of 1 XMR.").
  - _timestamp_ - unsigned int; The unix time at which the block was recorded into the blockchain.
  - _wide_cumulative_difficulty_ - Cumulative difficulty of all blocks in the blockchain as a hexadecimal string representing a 128-bit number.
  - _wide_difficulty_ - string; Network difficulty (analogous to the strength of the network) as a hexadecimal string representing a 128-bit number.
- _credits_ - unsigned int; If payment for RPC is enabled, the number of credits available to the requesting client. Otherwise, 0.

- _status_ - string; General RPC error code. "OK" means everything looks good.
- _top_hash_ - string; If payment for RPC is enabled, the hash of the highest block in the chain. Otherwise, empty.
- _untrusted_ - boolean; States if the result is obtained using the bootstrap mode, and is therefore not trusted (`true`), or when the daemon is fully synced and thus handles the RPC locally (`false`)

In this example, the most recent block (1562023 at the time) is returned:

```json
$ curl http://127.0.0.1:18081/json_rpc -d
'{"jsonrpc":"2.0","id":"0","method":"get_last_block_header"}' -H 'Content-Type:
application/json'

{
  "id": "0",
  "jsonrpc": "2.0",
  "result": {
    "block_header": {
      "block_size": 5500,
      "block_weight": 5500,
      "cumulative_difficulty": 86164894009456483,
      "cumulative_difficulty_top64": 0,
      "depth": 0,
      "difficulty": 227026389695,
      "difficulty_top64": 0,
      "hash": "a6ad87cf357a1aac1ee1d7cb0afa4c2e653b0b1ab7d5bf6af310333e43c59dd0",
      "height": 2286454,
      "long_term_weight": 5500,
      "major_version": 14,
      "miner_tx_hash": "a474f87de1645ff14c5e90c477b07f9bc86a22fb42909caa0705239298da96d0",
      "minor_version": 14,
      "nonce": 249602367,
      "num_txes": 3,
      "orphan_status": false,
      "pow_hash": "",
      "prev_hash": "fa17fefe1d05da775a61a3dc33d9e199d12af167ef0ab37e52b51e8487b50f25",
      "reward": 1181337498013,
      "timestamp": 1612088597,
      "wide_cumulative_difficulty": "0x1321e83bb8af763",
      "wide_difficulty": "0x34dbd3cabf"
    },
    "credits": 0,
    "status": "OK",
    "top_hash": "",
    "untrusted": false
  }
}
```

### **get_miner_data**

Provide the necessary data to create a custom block template. They are used by p2pool.

Inputs: _None_ .

Outputs:

- _major_version_ - unsigned int; major fork version.
- _height_ - unsigned int; current blockheight.
- _prev_id_ - string; previous block id.
- _seed_hash_ - string; RandomX seed hash.
- _difficulty_ - unsigned int; network. difficulty.
- _median_weight_ - unsigned int; median block weight.
- _already_generated_coins_ - unsigned int; coins mined by the network so far.
- _status_ - string; General RPC error code. "OK" means everything looks good.
- _tx_backlog_ - array of mineable mempool transactions.

  - _id_ - string;
  - _weight_ - unsigned int;
  - _fee_ - unsigned int;
- _untrusted_ - boolean; States if the result is obtained using the bootstrap mode, and is therefore not trusted (`true`), or when the daemon is fully synced and thus handles the RPC locally (`false`)

Example:

```json
$ curl http://127.0.0.1:18081/json_rpc -d
'{"jsonrpc":"2.0","id":"0","method":"get_miner_data"}' -H 'Content-Type: application/json'

{
  "id": "0",
  "jsonrpc": "2.0",
  "result": {
    "already_generated_coins": 18186022843595960691,
    "difficulty": "0x48afae42de",
    "height": 2731375,
    "major_version": 16,
    "median_weight": 300000,
    "prev_id": "78d50c5894d187c4946d54410990ca59a75017628174a9e8c7055fa4ca5c7c6d",
    "seed_hash": "a6b869d50eca3a43ec26fe4c369859cf36ae37ce6ecb76457d31ffeb8a6ca8a6",
    "status": "OK",
    "tx_backlog": [{
      "fee": 30700000,
      "id": "9868490d6bb9207fdd9cf17ca1f6c791b92ca97de0365855ea5c089f67c22208",
      "weight": 1535
    },{
      "fee": 44280000,
      "id": "b6000b02bbec71e18ad704bcae09fb6e5ae86d897ced14a718753e76e86c0a0a",
      "weight": 2214
    }],
    "untrusted": false
  }
}
```

### **get_output_distribution**

Get the number of transaction outputs per-block for given cleartext amounts (0 meaning RingCT) within a specified range.

If the distribution is "cumulative", then each value in the array represents the number of transaction outputs in that block plus all total outputs of that amount before that block.

Alias: _None_ .

Inputs:

- _amounts_ - array of unsigned int; cleartext amounts to look for (0 gets all RingCT outputs).
- _cumulative_ - boolean; (optional, default is `false`) States if the result should be cumulative (`true`) or not (`false`).
- _from_height_ - unsigned int; (optional, default is 0) Starting height to check from, inclusive.
- _to_height_ - unsigned int; (optional, default is 0) Ending height to check up to, inclusive. Set to 0 to get entire chain after _from_height_.
- _binary_ - boolean; for disabling epee encoding
- _compress_ - boolean; ignored if `binary` set to `false`

Outputs:

- _distributions_ - array of structure distribution as follows:
  - _amount_ - unsigned int

  - _base_ - unsigned int; The total number of outputs of *amount* in the chain before, not including, the block at _start_height_.
  - _distribution_ - array of unsigned int
  - _start_height_ - unsigned int; Note that this is not necessarily equal to *from_height*, especially for *amount*=`0` where *start_height* will be no less than the height of the v4 hardfork.
- _status_ - string; General RPC error code. "OK" means everything looks good.

Example:

```json
$ curl http://127.0.0.1:18081/json_rpc -d
'{"jsonrpc":"2.0","id":"0","method":"get_output_distribution","params":{"amounts":
[628780000],"from_height":1462078}}' -H 'Content-Type: application/json'

{
  "id": "0",
  "jsonrpc": "2.0",
  "result": {
    "distributions": [{
      "amount": 2628780000,
      "base": 0,
      "distribution": [ ... ],
      "start_height": 1462078
    }],
    "status": "OK"
  }
}
```

### **get_output_histogram**

Get a histogram of output amounts. For all amounts (possibly filtered by parameters), gives the number of outputs on the chain for that amount.
RingCT outputs counts as 0 amount.

Inputs:

- _amounts_ - list of unsigned int
- _min_count_ - unsigned int
- _max_count_ - unsigned int
- _unlocked_ - boolean
- _recent_cutoff_ - unsigned int

Outputs:

- _credits_ - unsigned int; If payment for RPC is enabled, the number of credits available to the requesting client. Otherwise, 0.
- _histogram_ - list of histogram entries, in the following structure:
  - _amount_ - unsigned int; Output amount in [atomic-units](https://www.getmonero.org/resources/moneropedia/atomic-units.html "Atomic Units refer to the smallest fraction of 1 XMR.")
  - _total_instances_ - unsigned int;
  - _unlocked_instances_ - unsigned int;
  - _recent_instances_ - unsigned int;
- _status_ - string; General RPC error code. "OK" means everything looks good.
- _top_hash_ - string; If payment for RPC is enabled, the hash of the highest block in the chain. Otherwise, empty.
- _untrusted_ - boolean; States if the result is obtained using the bootstrap mode, and is therefore not trusted (`true`), or when the daemon is fully synced and thus handles the RPC locally (`false`)

Example:

```json
$ curl http://127.0.0.1:18081/json_rpc -d
'{"jsonrpc":"2.0","id":"0","method":"get_output_histogram","params":{"amounts":
```

[20000000000]}}' -H 'Content-Type: application/json'

```
{
  "id": "0",
  "jsonrpc": "2.0",
  "result": {
    "credits": 0,
    "histogram": [{
      "amount": 20000000000,
      "recent_instances": 0,
      "total_instances": 381477,
      "unlocked_instances": 0
    }],
    "status": "OK",
    "top_hash": "",
    "untrusted": false
  }
}
```

### **get_txpool_backlog**

Get all transaction pool backlog

Alias: _None_ .

Inputs: _None_ .

Outputs:

- _backlog_ : array of structures _tx_backlog_entry_ (in binary form):
  - _blob_size_ - unsigned int (in binary form)
  - _fee_ - unsigned int (in binary form)
  - _time_in_pool_ - unsigned int (in binary form)
- _status_ - string; General RPC error code. "OK" means everything looks good.
- _untrusted_ - boolean; States if the result is obtained using the bootstrap mode, and is therefore not trusted (`true`), or when the daemon is fully synced and thus handles the RPC locally (`false`)

Example:

```json
$ curl http://127.0.0.1:18081/json_rpc -d
'{"jsonrpc":"2.0","id":"0","method":"get_txpool_backlog"}' -H 'Content-Type:
application/json'

{
  "id": "0",
  "jsonrpc": "2.0",
  "result": {
    "backlog": "...Binary...",
    "status": "OK",
    "untrusted": false
  }
}
```

### **get_version**

Give the node current version

Alias: _None_ .

Inputs: _None_ .

Outputs:

- _current_height_ - unsigned int; blockheight.
- _hard_forks_ - array of unsigned int.
  - _height_ - unsigned int;
  - _hf_version_ - unsigned int; hard fork version.
- _release_ - boolean; States if the daemon software version corresponds to an official tagged release (`true`), or not (`false`)
- _status_ - string; General RPC error code. "OK" means everything looks good.
- _untrusted_ - boolean; States if the result is obtained using the bootstrap mode, and is therefore not trusted (`true`), or when the daemon is fully synced and thus handles the RPC locally (`false`)
- _version_ - unsigned int;

Example:

```json
$ curl http://127.0.0.1:18081/json_rpc -d
'{"jsonrpc":"2.0","id":"0","method":"get_version"}' -H 'Content-Type: application/json'

{
  "id": "0",
  "jsonrpc": "2.0",
  "result": {
    "current_height": 3019446,
    "hard_forks": [{
      "height": 1,
      "hf_version": 1
    } ... {
      "height": 2689608,
      "hf_version": 16
    }],
    "release": true,
    "status": "OK",
    "untrusted": false,
    "version": 196613
  }
}
```

### **hard_fork_info**

Look up information regarding hard fork voting and readiness.

Alias: _None_ .

Inputs: _None_ .

Outputs:

- _credits_ - unsigned int; If payment for RPC is enabled, the number of credits available to the requesting client. Otherwise, 0.
- _earliest_height_ - unsigned int; Block height at which hard fork would be enabled if voted in.
- _enabled_ - boolean; Tells if hard fork is enforced.
- _state_ - unsigned int; Current hard fork state: 0 (There is likely a hard fork), 1 (An update is needed to fork properly), or 2 (Everything looks good).
- _status_ - string; General RPC error code. "OK" means everything looks good.
- _threshold_ - unsigned int; Minimum percent of votes to trigger hard fork. Default is 80.
- _top_hash_ - string; If payment for RPC is enabled, the hash of the highest block in the chain. Otherwise, empty.
- _$1_ - boolean; States if the result is obtained using the bootstrap mode, and is therefore not trusted (`true`), or when the daemon is fully synced and thus handles the RPC locally (`false`)
- _version_ - unsigned int; The major block version for the fork.
- _votes_ - unsigned int; Number of votes towards hard fork.

- _voting_ - unsigned int; Hard fork voting status.
- _window_ - unsigned int; Number of blocks over which current votes are cast. Default is 10080 blocks.

Example:

```json
$ curl http://127.0.0.1:18081/json_rpc -d
'{"jsonrpc":"2.0","id":"0","method":"hard_fork_info"}' -H 'Content-Type: application/json'

{
  "id": "0",
  "jsonrpc": "2.0",
  "result": {
    "credits": 0,
    "earliest_height": 2210720,
    "enabled": true,
    "state": 2,
    "status": "OK",
    "threshold": 0,
    "top_hash": "",
    "untrusted": false,
    "version": 14,
    "votes": 10080,
    "voting": 14,
    "window": 10080
  }
}
```

### **on_get_block_hash**

Look up a block's hash by its height.

Alias: _on_getblockhash_ .

Inputs:

* block height (int array of length 1)

Outputs:

* block hash (string)

Example:

```json
$ curl http://127.0.0.1:18081/json_rpc -d
'{"jsonrpc":"2.0","id":"0","method":"on_get_block_hash","params":[912345]}' -H 'Content-
Type: application/json'

{
  "id": "0",
  "jsonrpc": "2.0",
  "result": "e22cf75f39ae720e8b71b3d120a5ac03f0db50bba6379e2850975b4859190bc6"
}
```

### **prune_blockchain**

Alias: _None_ .

Inputs:

- _check_ - boolean; Optional (`false` by default) - If set to `true` then pruning status

is checked instead of initiating pruning.

Outputs:

- _pruned_ - boolean;
- _pruning_seed_ - unsigned int; Blockheight at which pruning began.
- _status_ - string; General RPC error code. "OK" means everything looks good.
- _untrusted_ - boolean; States if the result is obtained using the bootstrap mode, and is therefore not trusted (`true`), or when the daemon is fully synced and thus handles the RPC locally (`false`)

Example:

```json
$ curl http://127.0.0.1:18081/json_rpc -d
'{"jsonrpc":"2.0","id":"0","method":"prune_blockchain","params":{"check":true}}' -H
'Content-Type: application/json'

{
  "id": "0",
  "jsonrpc": "2.0",
  "result": {
    "pruned": true,
    "pruning_seed": 387,
    "status": "OK",
    "untrusted": false
  }
}
```

### **relay_tx**

Relay a list of transaction IDs.

Alias: _None_ .

Inputs:

- _txids_ - array of string; list of transaction IDs to relay

Outputs:

- _status_ - string; General RPC error code. "OK" means everything looks good.

Example:

```json
$ curl http://127.0.0.1:18081/json_rpc -d
'{"jsonrpc":"2.0","id":"0","method":"relay_tx","params":{"txids":
["9fd75c429cbe52da9a52f2ffc5fbd107fe7fd2099c0d8de274dc8a67e0c98613"]}}' -H 'Content-Type:
application/json'

{
  "id": "0",
  "jsonrpc": "2.0",
  "result": {
    "status": "OK"
  }
}
```

### **set_bans**

Ban another node by IP.

Alias: _None_ .

Inputs:

- _bans_ - A list of nodes to ban:
  - _host_ - string; Host to ban (IP in A.B.C.D form - will support I2P address in the future).
  - _ip_ - unsigned int; IP address to ban, in Int format.
  - _ban_ - boolean; Set `true` to ban.
  - _seconds_ - unsigned int; Number of seconds to ban node.

Outputs:

- _status_ - string; General RPC error code. "OK" means everything looks good.

Examples:

**banning by host**

In the following example, host is banned with its IP address string-formatted as A.B.C.D:

```json
$ curl http://127.0.0.1:18081/json_rpc -d
'{"jsonrpc":"2.0","id":"0","method":"set_bans","params":{"bans":
[{"host":"192.168.1.51","ban":true,"seconds":30}]}}' -H  'Content-Type: application/json'

{
  "id": "0",
  "jsonrpc": "2.0",
  "result": {
    "status": "OK",
    "untrusted": false
  }
}
```

**banning by ip**

In the following example, integer-formatted IP is banned:

```json
$ curl http://127.0.0.1:18081/json_rpc -d
'{"jsonrpc":"2.0","id":"0","method":"set_bans","params":{"bans":
[{"ip":838969536,"ban":true,"seconds":30}]}}' -H  'Content-Type: application/json'

{
  "id": "0",
  "jsonrpc": "2.0",
  "result": {
    "status": "OK"
  }
}
```

### **submit_block**

Submit a mined block to the network.

Alias: _submitblock_ .

Inputs:

* Block blob data - array of strings; list of block blobs which have been mined.  See [get_block_template](#get_block_template) to get a blob on which to mine.

Outputs:

- _status_ - string; Block submit status.

In this example, a block blob which has not been mined is submitted:

```json
$ curl http://127.0.0.1:18081/json_rpc -d
'{"jsonrpc":"2.0","id":"0","method":"submit_block","params":
["0707e6bdfedc053771512f1bc27c62731ae9e8f2443db64ce742f4e57f5cf8d393de28551e441a0000000002
fb830a01ffbf830a018cfe88bee283060274c0aae2ef5730e680308d9c00b6da59187ad0352efe3c71d36eeeb2
8782f29f2501bd56b952c3ddc3e350c2631d3a5086cac172c56893831228b17de296ff4669de020200000000"]
}' -H 'Content-Type: application/json'

{
  "error": {
    "code": -7,
    "message": "Block not accepted"
  },
  "id": "0",
  "jsonrpc": "2.0"
}
```

### **sync_info**

Get synchronization information

Alias: _None_ .

Inputs: _None_ .

Outputs:

- _credits_ - unsigned int; If payment for RPC is enabled, the number of credits available to the requesting client. Otherwise, 0.
- _height_ - unsigned int;
- _next_needed_pruning_seed_ - unsigned int; The next pruning seed needed for pruned sync.
- _overview_ - string; Overview of current block queue where each character in the string represents a block set in the queue. `. = requested but not received`, `o = set received`, `m  = received set that matches the next blocks needed`
- _peers_ - array of peer structure, defined as follows:
  - _info_ - structure of connection info, as defined in [get_connections](#get_connections)
- _spans_ - array of span structure, defined as follows (optional, absent if node is fully synced):
  - _connection_id_ - string; Id of connection
  - _nblocks_ - unsigned int; number of blocks in that span
  - _rate_ - unsigned int; connection rate
  - _remote_address_ - string; peer address the node is downloading (or has downloaded) than span from
  - _size_ - unsigned int; total number of bytes in that span's blocks (including txes)
  - _speed_ - unsigned int; connection speed
  - _start_block_height_ - unsigned int; block height of the first block in that span
- _status_ - string; General RPC error code. "OK" means everything looks good.
- _target_height_ - unsigned int; target height the node is syncing from (will be 0 if node is fully synced)

Example:

```json
$ curl http://127.0.0.1:18081/json_rpc -d
'{"jsonrpc":"2.0","id":"0","method":"sync_info"}' -H 'Content-Type: application/json'

{
  "id": "0",
  "jsonrpc": "2.0",
  "result": {
    "credits": 0,
    "height": 2287210,
```

```
    "next_needed_pruning_seed": 0,
    "overview": "[]",
    "peers": [{
      "info": {
        "address": "51.79.49.41:44317",
        "address_type": 1,
        "avg_download": 0,
        "avg_upload": 1,
        "connection_id": "718a970773e844618f3b830aa5775a45",
        "current_download": 0,
        "current_upload": 1,
        "height": 1,
        "host": "51.79.49.41",
        "incoming": true,
        "ip": "51.79.49.41",
        "live_time": 26,
        "local_ip": false,
        "localhost": false,
        "peer_id": "c1d50bcd29c89909",
        "port": "44317",
        "pruning_seed": 0,
        "recv_count": 468,
        "recv_idle_time": 5,
        "rpc_credits_per_hash": 0,
        "rpc_port": 0,
        "send_count": 35347,
        "send_idle_time": 3,
        "state": "normal",
        "support_flags": 1
      }
    },{

      "info": {
        ...
      }
    },{
      ...
    },{
      ...
    },{
      ...
    }],
    "status": "OK",
    "target_height": 2287203,
    "top_hash": "",
    "untrusted": false
  }
}
```


---

## Other Daemon RPC Calls

Not all daemon RPC calls use the JSON_RPC interface. This section gives examples of these calls.

The data structure for these calls is different than the JSON RPC calls. Whereas the JSON RPC methods were called using the `/json_rpc` extension and specifying a method, these methods are called at their own extensions. For example:

Note: It is recommended to use JSON RPC where such alternatives exist, rather than the following methods. For example, the recommended way to get a node's height is via the JSON RPC methods [get_info](#get_info) or [get_last_block_header](#get_last_block_header), rather than [get_height](#get_height) below.

For calls that end with **.bin**, the data is exchanged in the form of binary, serialized objects, as defined in the [Core RPC Server](https://github.com/monero-project/monero/blob/master/src/rpc/core_rpc_server_commands_defs.h).

### **/get_alt_blocks_hashes**

Get the known blocks hashes which are not on the main chain.

Alias: _None_ .

Inputs: _None_

Outputs:

- _blks_hashes_ - array of strings; list of alternative blocks hashes to main chain
- _credits_ - unsigned int; If payment for RPC is enabled, the number of credits available to the requesting client. Otherwise, 0.
- _status_ - string; General RPC error code. "OK" means everything looks good.
- _top_hash_ - string; If payment for RPC is enabled, the hash of the highest block in the chain. Otherwise, empty.
- _untrusted_ - boolean; States if the result is obtained using the bootstrap mode, and is therefore not trusted (`true`), or when the daemon is fully synced and thus handles the RPC locally (`false`)

Example:

```json
$ curl http://127.0.0.1:18081/get_alt_blocks_hashes -H 'Content-Type: application/json'

{
  "blks_hashes":
["dd4998cfe92a959a5a0e4ed72432cf23d7dfc4179cbea871ee2a705d71fb5e25","f36c3856ffde6a7d06fc8
32c80ede4ad5b6c8f702c9736dae1e2140d86504db9","8d0c1f806817259d213c8829ea3356334e0d8fdd3b11
8e1243756e12dce767bb"],
  "credits": 0,
  "status": "OK",
  "top_hash": "",
    "untrusted": false
    "untrusted": false
  }
  "untrusted": false
  }
}
```



### **/get_blocks.bin**

Get all blocks info. Binary request.

Alias: _/getblocks.bin_ .

Inputs:

- _block_ids_ - binary array of hashes; first 10 blocks id goes sequential, next goes in pow(2,n) offset, like 2, 4, 8, 16, 32, 64 and so on, and the last one is always genesis block
- _start_height_ - unsigned int
- _prune_ - boolean

Outputs:

- _blocks_ - array of block complete entries
- _current_height_ - unsigned int
- _output_indices_ - structure as follows:
  - _indices_ - array of tx output indices, structure as follows:
    - _indices_ - array of unsigned int
- _start_height_ - unsigned int
- _status_ - string; General RPC error code. "OK" means everything looks good.
- _untrusted_ - boolean; States if the result is obtained using the bootstrap mode, and is

therefore not trusted (`true`), or when the daemon is fully synced and thus handles the RPC locally (`false`)

<!-- Cannot get this working
Example:

```json
$ curl http://127.0.0.1:18081/get_blocks.bin -d '{"block_ids":
["d109a406528a7b44fef8bc03e75eaabb0f919f852884b43b550b8b3be80a49e7"],"start_height":156206
2}' -H 'Content-Type: application/json'

```
--->


### **/get_blocks_by_height.bin**

Get blocks by height. Binary request.

Alias: _/getblocks_by_height.bin_ .

Inputs:

- _heights_ - array of unsigned int; list of block heights

Outputs:

- _blocks_ - array of block complete entries
- _status_ - string; General RPC error code. "OK" means everything looks good.
- _untrusted_ - boolean; States if the result is obtained using the bootstrap mode, and is therefore not trusted (`true`), or when the daemon is fully synced and thus handles the RPC locally (`false`)

<!-- Cannot get this working
Example:

```Bash
$ echo -e
'\x7B\x22\x68\x65\x69\x67\x68\x74\x73\x22\x3A\x5B\x31\x35\x36\x34\x32\x34\x36\x5D\x7D\x' |
curl --data-binary @- http://127.0.0.1:18081/get_blocks_by_height.bin
$ echo -e '1564246' | curl --data-binary @-
http://127.0.0.1:18081/get_blocks_by_height.bin
curl http://127.0.0.1:18081/get_blocks_by_height.bin --data-binary '{"heights":[1564246]}'

```
--->



### **/get_hashes.bin**

Get hashes. Binary request.

Alias: _/gethashes.bin_ .

Inputs:

- _block_ids_ - binary array of hashes; first 10 blocks id goes sequential, next goes in pow(2,n) offset, like 2, 4, 8, 16, 32, 64 and so on, and the last one is always genesis block
- _start_height_ - unsigned int

Outputs:

- _current_height_ - unsigned int
- _m_block_ids_ - binary array of hashes; see _block_ids_ above.
- _start_height_ - unsigned int
- _status_ - string; General RPC error code. "OK" means everything looks good.
- _untrusted_ - boolean; States if the result is obtained using the bootstrap mode, and is

therefore not trusted (`true`), or when the daemon is fully synced and thus handles the RPC locally (`false`)

<!-- Cannot get this working
Example:

```json
$ curl http://127.0.0.1:18081/get_hashes.bin -d '{"block_ids":
["d109a406528a7b44fef8bc03e75eaabb0f919f852884b43b550b8b3be80a49e7"],"start_height":156206
2}' -H 'Content-Type: application/json'

```
--->

### **/get_height**

Get the node's current height.

Alias: _/getheight_ .

Inputs: _None_ .

Outputs:

- _hash_ - string; The block's hash.
- _height_ - unsigned int; Current length of longest chain known to daemon.
- _status_ - string; General RPC error code. "OK" means everything looks good.
- _untrusted_ - boolean; States if the result is obtained using the bootstrap mode, and is therefore not trusted (`true`), or when the daemon is fully synced and thus handles the RPC locally (`false`)

```json
$ curl http://127.0.0.1:18081/get_height -H 'Content-Type: application/json'

{
  "hash": "7e23a28cfa6df925d5b63940baf60b83c0cbb65da95f49b19e7cf0ce7dd709ce",
  "height": 2287217,
  "status": "OK",
  "untrusted": false
}
```

### **/get_info (not JSON)**

This method is a convenient backward support and should not be used anymore. See [get_info](#get_info) JSON RPC for details.

Alias:

- _/getinfo_
- _get_info_

### **/get_limit**

Get daemon bandwidth limits.

Alias: _None_ .

Inputs: _None_ .

Outputs:

- _limit_down_ - unsigned int; Download limit in kBytes per second
- _limit_up_ - unsigned int; Upload limit in kBytes per second
- _status_ - string; General RPC error code. "OK" means everything looks good.
- _untrusted_ - boolean; States if the result is obtained using the bootstrap mode, and is therefore not trusted (`true`), or when the daemon is fully synced and thus handles the RPC locally (`false`)

Example:

```json
$ curl http://127.0.0.1:18081/get_limit -H 'Content-Type: application/json'

{
  "limit_down": 8192,
  "limit_up": 128,
  "status": "OK",
  "untrusted": false
}
```

### **/get_net_stats**

Alias: _None_ .

Inputs: _None_ .

Outputs:

- _start_time_ - unsigned int; Unix start time.
- _total_packets_in_ - unsigned int;
- _total_bytes_in_ - unsigned int;
- _total_packets_out_ - unsigned int;
- _total_bytes_out_ - unsigned int;
- _status_ - string; General RPC error code. "OK" means everything looks good.
- _untrusted_ - boolean; States if the result is obtained using the bootstrap mode, and is therefore not trusted (`true`), or when the daemon is fully synced and thus handles the RPC locally (`false`).

Example:

```json
$ curl http://127.0.0.1:18081/get_net_stats -H 'Content-Type: application/json'

{
  "start_time": 1665147355,
  "status": "OK",
  "total_bytes_in": 3743474809,
  "total_bytes_out": 5932012405,
  "total_packets_in": 2130592,
  "total_packets_out": 1010674,
  "untrusted": false
}
```

### **/get_o_indexes.bin**

Get global outputs of transactions. Binary request.

Alias: _None_ .

Inputs:

- _txid_ - binary txid

Outputs:

- _o_indexes_ - array of unsigned int; List of output indexes
- _status_ - string; General RPC error code. "OK" means everything looks good.
- _untrusted_ - boolean; States if the result is obtained using the bootstrap mode, and is therefore not trusted (`true`), or when the daemon is fully synced and thus handles the RPC locally (`false`)

<!-- Cannot get this working
Example:

```Bash
$ curl http://127.0.0.1:18081/get_o_indexes.bin --data-binary
'{"txid":"d6e48158472848e6687173a91ae6eebfa3e1d778e65252ee99d7515d63090408"}'

```
--->

### **/get_output_distribution.bin**

Get outputs. Binary request version of [get_output_distribution]
(#get_output_distribution).

Alias: _None_ .

Inputs:

- _credits_ - unsigned int; If payment for RPC is enabled, the number of credits available to the requesting client. Otherwise, 0.
- _top_hash_ - string; If payment for RPC is enabled, the hash of the highest block in the chain. Otherwise, empty.
- _amounts_ - array of unsigned int; amounts to look for
- _cumulative_ - boolean; (optional, default is `false`) States if the result should be cumulative (`true`) or not (`false`)
- _from_height_ - unsigned int; (optional, default is 0) starting height to check from
- _to_height_ - unsigned int; (optional, default is 0) ending height to check up to
- _binary_ - boolean; (optional, default is `true`); Must be `true`.
- _compress_ - unsigned int; (optional, default is `false`) Respond with compressed data.

Outputs:

- _credits_ - unsigned int; If payment for RPC is enabled, the number of credits available to the requesting client. Otherwise, 0.
- _distributions_ - array of structure distribution as follows:
  - _amount_ - unsigned int
  - _base_ - unsigned int
  - _binary_ - boolean; Always `true`.
  - _compress_ - unsigned int; `true` if compressed data, `false` otherwise.
  - _distribution_ - array of unsigned int; This field exists instead of _compressed_data_ if the _compress_ option was not used.
  - _compressed_data_ - array of unsigned int; This field exists instead of _distribution_ if the _compress_ option was used.
  - _start_height_ - unsigned int
- _status_ - string; General RPC error code. "OK" means everything looks good.
- _top_hash_ - string; If payment for RPC is enabled, the hash of the highest block in the chain. Otherwise, empty.
- _untrusted_ - boolean; States if the result is obtained using the bootstrap mode, and is therefore not trusted (`true`), or when the daemon is fully synced and thus handles the RPC locally (`false`)

### **/get_outs**

Get outputs.

Alias: _None_ .

Inputs:

- _outputs_ array of _get_outputs_out_ structure as follows:
  - _amount_ - unsigned int;
  - _index_ - unsigned int;
- _get_txid_ - boolean; If `true`, a _txid_ will included for each output in the response.

Outputs:

- _outs_ - array of structure _outkey_ as follows:
  - _height_ - unsigned int; block height of the output
  - _key_ - String; the public key of the output
  - _mask_ - String
  - _txid_ - String; transaction id
  - _unlocked_ - boolean; States if output is locked (`false`) or not (`true`)
- _status_ - string; General RPC error code. "OK" means everything looks good.
- _untrusted_ - boolean; States if the result is obtained using the bootstrap mode, and is therefore not trusted (`true`), or when the daemon is fully synced and thus handles the RPC locally (`false`)

### **/get_outs.bin**

Get outputs. Binary request.

Alias: _None_ .

Inputs:

- _outputs_ - array of structure _get_outputs_out_ as follows:
  - _amount_ - unsigned int;
  - _index_ - unsigned int;

Outputs:

- _outs_ - array of structure _outkey_ as follows:
  - _amount_ - unsigned int;
  - _height_ - unsigned int; block height of the output
  - _key_ - the public key of the output
  - _mask_
  - _txid_ - transaction id
  - _unlocked_ - boolean; States if output is locked (`false`) or not (`true`)
- _status_ - string; General RPC error code. "OK" means everything looks good.
- _untrusted_ - boolean; States if the result is obtained using the bootstrap mode, and is therefore not trusted (`true`), or when the daemon is fully synced and thus handles the RPC locally (`false`)

<!-- Cannot get this working
Example:

```Bash
$ curl http://127.0.0.1:18081/get_o_indexes.bin --data-binary
'{"txid":"d6e48158472848e6687173a91ae6eebfa3e1d778e65252ee99d7515d63090408"}'

```
--->

### **/get_peer_list**

Get the known peers list.

Alias: _None_ .

Inputs: _None_ .

Outputs:

- _gray_list_ - array of offline _peer_ structure as follows:
  - _host_ - unsigned int; IP address in integer format
  - _id_ - string; Peer id
  - _ip_ - unsigned int; IP address in integer format
  - _last_seen_ - unsigned int; unix time at which the peer has been seen for the last time
  - _port_ - unsigned int; TCP port the peer is using to connect to monero network.
- _status_ - string; General RPC error code. "OK" means everything looks good. Any other value means that something went wrong.
- _white_list_ - array of online _peer_ structure, as above.

Example (truncated lists):

```json
$ curl http://127.0.0.1:18081/get_peer_list -H 'Content-Type: application/json'

{
  "gray_list": [{
    "host": "640304833",
    "id": 5345237316225602120,
    "ip": 640304833,
    "last_seen": 1525540510,
    "port": 18080
  },{
    "host": "2183731038",
    "id": 14955030573998424430,
    "ip": 2183731038,
    "last_seen": 1525540499,
    "port": 28080
  }, ...
  ],
  "status": "OK",
  "white_list": [{
    "host": "1221637955",
    "id": 10354694710033118926,
    "ip": 1221637955,
    "last_seen": 1525540511,
    "port": 18080
  },{
    "host": "1780407354",
    "id": 17193661050352240890,
    "ip": 1780407354,
    "last_seen": 1525540510,
    "port": 18080
  }, ...
  ]
}
```

### **/get_public_nodes**

Get public peer information.

Alias: _None_ .

Inputs:

- _gray_ - boolean; (Optional; defaults to `false`) Include gray peers.
- _white_ - boolean; (Optional; defaults to `true`) Include white peers.
- _include_blocked_ - boolean; (Optional; defaults to `false`) Include blocked peers.

Outputs:

- _status_ - string; General RPC error code. "OK" means everything looks good.
- _whites_ - array of `public_node` structures defined as follows:
  - _host_ - string; The node's IP address. This includes IPv4, IPv6, Onion, and i2p

addresses.
  - _last_seen_ - unsigned int; UNIX timestamp of the last time the node was seen.
  - _rpc_credits_per_hash_ - unsigned int; If payment for RPC is enabled, the number of credits the node is requesting per hash. Otherwise, 0.
  - _rpc_port_ - unsigned int; RPC port number of the node.
- _gray_ - array of `public_node` structures; This is an array structures containing node information on gray peers. See _white_ above.
- _untrusted_ - boolean; States if the result is obtained using the bootstrap mode, and is therefore not trusted (`true`), or when the daemon is fully synced and thus handles the RPC locally (`false`)

Example:

```json
$ curl http://127.0.0.1:18081/get_public_nodes -d '{"gray":true}' -H 'Content-Type: application/json'

{
  "gray": [{
    "host": "142.93.132.1",
    "last_seen": 0,
    "rpc_credits_per_hash": 0,
    "rpc_port": 18089
  },{
    "host": "193.142.4.2",
    "last_seen": 0,
    "rpc_credits_per_hash": 0,
    "rpc_port": 19084
  }],
  "status": "OK",
  "untrusted": false,
  "white": [{
    "host": "70.52.75.3",
    "last_seen": 1722812577,
    "rpc_credits_per_hash": 0,
    "rpc_port": 18081
  },{
    "host": "::ffff:207.180.221.220",
    "last_seen": 1722872337,
    "rpc_credits_per_hash": 0,
    "rpc_port": 18089
  },{
    "host": "zbjkbsxc5munw3qusl7j2hpcmikhqocdf4pqhnhtpzw5nt5jrmofptid.onion:18083",
    "last_seen": 1720186288,
    "rpc_credits_per_hash": 0,
    "rpc_port": 18089
  }]
}
```
### **/get_transaction_pool**

Show information about valid transactions seen by the node but not yet mined into a block, as well as spent key image information for the txpool in the node's memory.

Alias: _None_ .

Inputs: _None_ .

Outputs:

- _credits_ - unsigned int; If payment for RPC is enabled, the number of credits available to the requesting client. Otherwise, 0.
- _spent_key_images_ - List of spent output key images:
  - _id_hash_ - string; Key image.
  - _txs_hashes_ - string list; tx hashes of the txes (usually one) spending that key image.
- _status_ - string; General RPC error code. "OK" means everything looks good.
- _transactions_ - List of transactions in the mempool are not in a block on the main chain at the moment:

  - _blob_size_ - unsigned int; The size of the full transaction blob.
  - _do_not_relay_ ; boolean; States if this transaction should not be relayed
  - _double_spend_seen_ - boolean; States if this transaction has been seen as double spend.
  - _fee_ - unsigned int; The amount of the mining fee included in the transaction, in [atomic-units](https://www.getmonero.org/resources/moneropedia/atomic-units.html "Atomic Units refer to the smallest fraction of 1 XMR.").
  - _id_hash_ - string; The transaction ID hash.
  - _kept_by_block_ - boolean; States if the tx was included in a block at least once (`true`) or not (`false`).
  - _last_failed_height_ - unsigned int; If the transaction validation has previously failed, this tells at what height that occurred.
  - _last_failed_id_hash_ - string; Like the previous, this tells the previous transaction ID hash.
  - _last_relayed_time_ - unsigned int; Last unix time at which the transaction has been relayed.
  - _max_used_block_height_ - unsigned int; Tells the height of the most recent block with an output used in this transaction.
  - _max_used_block_id_hash_ - string; Tells the hash of the most recent block with an output used in this transaction.
  - _receive_time_ - unsigned int; The Unix time that the transaction was first seen on the network by the node.
  - _relayed_ - boolean; States if this transaction has been relayed
  - _tx_blob_ - unsigned int; Hexadecimal blob representing the transaction.
  - _tx_json_ - json string; JSON structure of all information in the transaction:
    - _version_ - Transaction version
    - _unlock_time_ - If not 0, this tells when a transaction output is spendable.
    - _vin_ - List of inputs into transaction:
      - _key_ - The public key of the previous output spent in this transaction.
        - _amount_ - The amount of the input, in [atomic-units](https://www.getmonero.org/resources/moneropedia/atomic-units.html "Atomic Units refer to the smallest fraction of 1 XMR.").
        - _key_offsets_ - A list of integer offsets to the input.
        - _k_image_ - The key image for the given input
    - _vout_ - List of outputs from transaction:
      - _amount_ - Amount of transaction output (if coinbase output, otherwise 0), in [atomic-units](https://www.getmonero.org/resources/moneropedia/atomic-units.html "Atomic Units refer to the smallest fraction of 1 XMR.").
      - _target_ - Output destination information:
        - _tagged_key_
          - _key_ - The stealth public key of the receiver. Whoever owns the private key associated with this key controls this transaction output.
          - _view_tag_ - The 1st byte of a shared secret (used for reducing synchronization time).
    - _extra_ - Usually called the "transaction ID" but can be used to include any random 32 bytes.
    - _rct_signatures_ - List of signatures used in ring signature to hide the true origin of the transaction.
      - _ecdhInfo_ - array of Diffie Helman Elipctic curves structures as follows:
        - _mask_ - String;
        - _amount_ - String;
      - _outPk_ - List
      - _txnFee_ -
      - _type_ -
    - _rctsig_prunable_ -
      - _CLSAGs_ - List
        - _D_ -
        - _c1_ -
        - _s_ - List
      - _bpp_ - List
        - _A_ -
        - _A1_ -
        - _B_ -
        - _L_ - List
        - _R_ - List
        - _d1_ -
        - _r1_ -
        - _s1_ -
      - _npb_ -

      - _pseudoOuts_ -
  - _weight_ -

Example (Note: Some lists in the returned information have been truncated for display reasons):

```json
$ curl http://127.0.0.1:18081/get_transaction_pool -H 'Content-Type: application/json'

{
  "spent_key_images": [{
    "id_hash": "5b5d68e7eb5ddfc7dd30d9a91f1db6835563b0214b66dc0b791909fb46c9cbb4",
    "txs_hashes": ["0b1839626bd7a609d5b16966396a51eb5fda9c55b9b2fc314a916459a7532e02"]
  },{
    "id_hash": "b583a134ccd6fd48697f02f530e82604f746b97ce26de103fa756d91274ac60e",
    "txs_hashes": ["0b1839626bd7a609d5b16966396a51eb5fda9c55b9b2fc314a916459a7532e02"]
  },
  ...],
  "status": "OK",
  "transactions": [{
    "blob_size": 2217,
    "do_not_relay": false,
    "double_spend_seen": false,
    "fee": 44340000,
    "id_hash": "0b1839626bd7a609d5b16966396a51eb5fda9c55b9b2fc314a916459a7532e02",
    "kept_by_block": false,
    "last_failed_height": 0,
    "last_failed_id_hash":
"0000000000000000000000000000000000000000000000000000000000000000",
    "last_relayed_time": 1667929927,
    "max_used_block_height": 2751399,
    "max_used_block_id_hash":
"52d2ac5230146c4d8003c8fb1d8e54aea91d2536c72884bd6e126b7def4ed9c6",
    "receive_time": 1667929927,
    "relayed": true,
    "tx_blob": " ... ",
    "tx_json": "{\n  \"version\": 2, \n  \"unlock_time\": 0, \n  \"vin\": [ {\n
\"key\": {\n        \"amount\": 0, \n        \"key_offsets\": [ 54937085, 875767, 6217742,
360564, 431709, 807289, 138472, 6221, 111408, 42881, 145, 12396, 20911, 899, 3546, 112\n
], \n        \"k_image\":
\"b583a134ccd6fd48697f02f530e82604f746b97ce26de103fa756d91274ac60e\"\n      }\n    }, {\n
\"key\": {\n        \"amount\": 0, \n        \"key_offsets\": [ 28132301, 33525919, 17414,
2042026, 73478, 73012, 54330, 8150, 5725, 10740, 4581, 7687, 9290, 592, 994, 725\n
], \n        \"k_image\":
\"5b5d68e7eb5ddfc7dd30d9a91f1db6835563b0214b66dc0b791909fb46c9cbb4\"\n      }\n    }\n  ],
\n  \"vout\": [ {\n      \"amount\": 0, \n      \"target\": {\n        \"tagged_key\": {\n
\"key\": \"2849325ea3410c1378ab631c4e008028a99843a767c1d11019631cad8e083b53\", \n
\"view_tag\": \"1f\"\n        }\n      }\n    }, {\n      \"amount\": 0, \n
\"target\": {\n        \"tagged_key\": {\n          \"key\":
\"9677c3568218bab3dc1a6eee5b486eeb372c9b2c7d552d8f9c6c84bd4051bab8\", \n
\"view_tag\": \"ee\"\n        }\n      }\n    }\n  ], \n  \"extra\": [ 1, 110, 21, 19, 40,
178, 62, 51, 216, 134, 71, 171, 159, 249, 160, 206, 100, 247, 190, 162, 177, 0, 35, 237,
165, 157, 42, 84, 52, 25, 151, 202, 231, 2, 9, 1, 195, 62, 127, 79, 4, 67, 161, 64\n  ],
\n  ... }\n}",
    "weight": 2217
  },
  ...]
}
```

### **/get_transaction_pool_hashes**

Get hashes from transaction pool.

Alias: _None_ .

Inputs: _None_ .

Outputs:

- _credits_ - unsigned int; If payment for RPC is enabled, the number of credits available to the requesting client. Otherwise, 0.
- _status_ - string; General RPC error code. "OK" means everything looks good.
- _top_hash_ - string; If payment for RPC is enabled, the hash of the highest block in the chain. Otherwise, empty.
- _tx_hashes_ - array of strings; This is an array of transaction hashes in hexadecimal string form.
- _untrusted_ - boolean; States if the result is obtained using the bootstrap mode, and is therefore not trusted (`true`), or when the daemon is fully synced and thus handles the RPC locally (`false`)

Example:

```json
$ curl http://127.0.0.1:18081/get_transaction_pool_hashes -H 'Content-Type:
application/json'

{
  "credits": 0,
  "status": "OK",
  "top_hash": "",
  "tx_hashes":
["e27bb84f1cf1f47a79578d3bb1f47a81745a4c5a5551622b4e924eb0b83f960d","b9f76e686f389e126ada5
caf47034c57adeef671d8b899cba1ead606c868f819"],
  "untrusted": false
}
```

### **/get_transaction_pool_hashes.bin**

Get hashes from transaction pool. Binary request.

Alias: _None_ .

Inputs: _None_ .

Outputs:

- _status_ - string; General RPC error code. "OK" means everything looks good.
- _tx_hashes_ - binary array of transaction hashes.
- _untrusted_ - boolean; States if the result is obtained using the bootstrap mode, and is therefore not trusted (`true`), or when the daemon is fully synced and thus handles the RPC locally (`false`)

Example:

```json
$ curl http://127.0.0.1:18081/get_transaction_pool_hashes.bin -H 'Content-Type:
application/json'

{
  "status": "OK",
  "tx_hashes": " ... ",
  "untrusted": false
}
```

### **/get_transaction_pool_stats**

Get the transaction pool statistics.

Alias: _None_ .

Inputs: _None_ .

Outputs:

- _credits_ - unsigned int; If payment for RPC is enabled, the number of credits available to the requesting client. Otherwise, 0.
- _pool_stats_ - Structure as follows:
  - _bytes_max_ - unsigned int; Max transaction size in pool
  - _bytes_med_ - unsigned int; Median transaction size in pool
  - _bytes_min_ - unsigned int; Min transaction size in pool
  - _bytes_total_ - unsigned int; total size of all transactions in pool
  - _fee_total_ - unsigned int; The sum of the fees for all transactions currently in the transaction pool [atomic-units](https://www.getmonero.org/resources/moneropedia/atomic-units.html "Atomic Units refer to the smallest fraction of 1 XMR.").
  - _histo_ - structure _txpool_histo_ as follows:
    - _txs_ - unsigned int; number of transactions
    - _bytes_ - unsigned int; size in bytes.
  - _histo_98pc_ unsigned int; the time 98% of txes are "younger" than
  - _num_10m_ unsigned int; number of transactions in pool for more than 10 minutes
  - _num_double_spends_ unsigned int; number of double spend transactions
  - _num_failing_ unsigned int; number of failing transactions
  - _num_not_relayed_ unsigned int; number of non-relayed transactions
  - _oldest_ unsigned int; unix time of the oldest transaction in the pool
  - _txs_total_ unsigned int; total number of transactions.
- _status_ - string; General RPC error code. "OK" means everything looks good.
- _top_hash_ - string; If payment for RPC is enabled, the hash of the highest block in the chain. Otherwise, empty.
- _untrusted_ - boolean; States if the result is obtained using the bootstrap mode, and is therefore not trusted (`true`), or when the daemon is fully synced and thus handles the RPC locally (`false`)

Example:

```json
$ curl http://127.0.0.1:18081/get_transaction_pool_stats -H 'Content-Type: application/json'

{
  "credits": 0,
  "pool_stats": {
    "bytes_max": 14325,
    "bytes_med": 1460,
    "bytes_min": 1450,
    "bytes_total": 410674,
    "fee_total": 18198895000,
    "histo": [{ ...
    }],
    "histo_98pc": 297,
    "num_10m": 1,
    "num_double_spends": 1,
    "num_failing": 0,
    "num_not_relayed": 0,
    "oldest": 1612232218,
    "txs_total": 124
  },
  "status": "OK",
  "top_hash": "",
  "untrusted": false

}
```

### **/get_transactions**

Look up one or more transactions by hash.

Alias: _/gettransactions_ .

Inputs:

- _txs_hashes_ - string list; List of transaction hashes to look up.
- _decode_as_json_ - boolean; Optional (`false` by default). If set `true`, the returned transaction information will be decoded rather than binary.
- _prune_ - boolean; Optional (`false` by default).
- _split_ - boolean; Optional (`false` by default).

Outputs:

- _missed_tx_ - array of strings. (Optional - returned if not empty) Transaction hashes that could not be found.
- _status_ - General RPC error code. "OK" means everything looks good.
- _top_hash_ - string; If payment for RPC is enabled, the hash of the highest block in the chain. Otherwise, empty.
- _txs_ - array of structure _entry_ as follows:
  - _as_hex_ - string; Full transaction information as a hex string.
  - _as_json_ - json string; List of transaction info:
    - _version_ - Transaction version
    - _unlock_time_ - If not 0, this tells when a transaction output is spendable.
    - _vin_ - List of inputs into transaction:
      - _key_ - The public key of the previous output spent in this transaction.
        - _amount_ - The amount of the input, in [atomic-units](https://www.getmonero.org/resources/moneropedia/atomic-units.html "Atomic Units refer to the smallest fraction of 1 XMR.").
        - _key_offsets_ - A list of integer offsets to the input.
        - _k_image_ - The key image for the given input
    - _vout_ - List of outputs from transaction:
      - _amount_ - Amount of transaction output (if coinbase output, otherwise 0), in [atomic-units](https://www.getmonero.org/resources/moneropedia/atomic-units.html "Atomic Units refer to the smallest fraction of 1 XMR.").
      - _target_ - Output destination information:
        - _tagged_key_
          - _key_ - The stealth public key of the receiver. Whoever owns the private key associated with this key controls this transaction output.
          - _view_tag_ - The 1st byte of a shared secret (used for reducing synchronization time).
    - _extra_ - Usually called the "payment ID" but can be used to include any random 32 bytes.
    - _rct_signatures_ - List of signatures used in ring signature to hide the true origin of the transaction.
      - _ecdhInfo_ - array of Diffie Helman Elipctic curves structures as follows:
        - _mask_ - String;
        - _amount_ - String;
      - _outPk_ - List
      - _txnFee_ -
      - _type_ -
    - _rctsig_prunable_ -
      - _CLSAGs_ - List
        - _D_ -
        - _c1_ -
        - _s_ - List
      - _bpp_ - List
        - _A_ -
        - _A1_ -
        - _B_ -
        - _L_ - List
        - _R_ - List
        - _d1_ -
        - _r1_ -
        - _s1_ -
      - _npb_ -
      - _pseudoOuts_ -
  - _block_height_ - unsigned int; block height including the transaction
  - _block_timestamp_ - unsigned int; Unix time at chich the block has been added to the blockchain
  - _double_spend_seen_ - boolean; States if the transaction is a double-spend (`true`) or

not (`false`)
  - _in_pool_ - boolean; States if the transaction is in pool (`true`) or included in a block (`false`)
  - _output_indices_ - array of unsigned int; transaction indexes
  - _prunable_as_hex_ - string; Prunable block encoded as a hex string.
  - _prunable_hash_ - string; Keccak-256 hash of the prunable portion of the block.
  - _pruned_as_hex_ - string; Pruned block encoded as hex string.
  - _tx_hash_ - string; transaction hash
- _txs_as_hex_ - string; Full transaction information as a hex string (old compatibility parameter)
- _txs_as_json_ - json string; (Optional - returned if set in inputs. Old compatibility parameter) List of transaction as in _as_json_ above:

Example 1: Return transaction information in binary format.

```json
$ curl http://127.0.0.1:18081/get_transactions -d '{"txs_hashes":
["d6e48158472848e6687173a91ae6eebfa3e1d778e65252ee99d7515d63090408"]}' -H 'Content-Type:
application/json'

{
  "credits": 0,
  "status": "OK",
  "top_hash": "",
  "txs": [{
    "as_hex": "...",
    "as_json": "",
    "block_height": 993442,
    "block_timestamp": 1457749396,
    "double_spend_seen": false,
    "in_pool": false,
    "output_indices": [198769,418598,176616,50345,509],
    "prunable_as_hex": "",
    "prunable_hash": "0000000000000000000000000000000000000000000000000000000000000000",
    "pruned_as_hex": "",
    "tx_hash": "d6e48158472848e6687173a91ae6eebfa3e1d778e65252ee99d7515d63090408"
  }],

  "txs_as_hex": ["..."],
  "untrusted": false
}
```

Example 2: Decode returned transaction information in JSON format. Note: the "vin", "vout", "rct_signatures", "rctsig_prunable" list have been truncated in the displayed return for space considerations.

```json
$ curl http://127.0.0.1:18081/get_transactions -d '{"txs_hashes":
["a9c71fe27ccf978a56ef96e50b680a3d569754cd4d070e51d10fa9f6f658b8e3"],"decode_as_json":true
}' -H 'Content-Type: application/json'

{
  "credits": 0,
  "status": "OK",
  "top_hash": "",
  "txs": [{
    "as_hex": "...",
    "as_json": "{\n  \"version\": 2, \n  \"unlock_time\": 0, \n  \"vin\": [ {\n
\"key\": {\n        \"amount\": 0, \n        \"key_offsets\": [ 10077101, 34287013,
18015596, 907866, 44197, 192564, 52932, 325, 253684, 12715, 11997, 53334, 1265, 38895,
2549, 4459\n        ], \n        \"k_image\":
\"0931bae7689a009d7ab3eb9e3f2a1bb22f3ca1951ff68a27150 6c12f619c23f7\"\n      }\n    }\n  ],
\n  \"vout\": [ {\n      \"amount\": 0, \n      \"target\": {\n        \"tagged_key\": {\n
\"key\": \"b49eda305377eb6bd14107e38b0e444219c57a34b47ed75b82bf616b4bcc0de2\", \n
\"view_tag\": \"44\"\n        }\n      }\n    }, ... \n      }\n    }\n  ], \n  \"extra\":
[ 1, 128, 11, 79, 9, 245, 251, 172, 229, 55, 56, 18, 217, 12, 137, 129, 115, 245, 29, 99,
20, 152, 197, 212, 113, 41, 183, 252, 160, 202, 147, 176, 12, 2, 9, 1, 252, 7, 158, 153,
11, 223, 205, 97\n  ], \n ... }\n}",
```

```
    "block_height": 2751288,
    "block_timestamp": 1667916404,
    "confirmations": 44,
    "double_spend_seen": false,
    "in_pool": false,
    "output_indices": [63958274,63958275],
    "prunable_as_hex": "",
    "prunable_hash": "cdc54f848f991f48cbb64d76617570ebbe9787bd9eff71b9e36af3d409686650",
    "pruned_as_hex": "",
    "tx_hash": "a9c71fe27ccf978a56ef96e50b680a3d569754cd4d070e51d10fa9f6f658b8e3"
  }],
  "txs_as_hex": ["..."],
  "txs_as_json": ["{\n  \"version\": 2, \n  \"unlock_time\": 0, \n  \"vin\": [ {\n
\"key\": {\n        \"amount\": 0, \n        \"key_offsets\": [ 10077101, 34287013,
18015596, 907866, 44197, 192564, 52932, 325, 253684, 12715, 11997, 53334, 1265, 38895,
2549, 4459\n        ], \n        \"k_image\":
\"0931bae7689a009d7ab3eb9e3f2a1bb22f3ca1951ff68a271506c12f619c23f7\"\n      }\n    }\n  ],
\n  \"vout\": [ {\n      \"amount\": 0, \n      \"target\": {\n        \"tagged_key\": {\n
\"key\": \"b49eda305377eb6bd14107e38b0e444219c57a34b47ed75b82bf616b4bcc0de2\", \n
\"view_tag\": \"44\"\n        }\n      }\n    }, ... \n      }\n    }\n  ], \n  \"extra\":
[ 1, 128, 11, 79, 9, 245, 251, 172, 229, 55, 56, 18, 217, 12, 137, 129, 115, 245, 29, 99,
20, 152, 197, 212, 113, 41, 183, 252, 160, 202, 147, 176, 12, 2, 9, 1, 252, 7, 158, 153,
11, 223, 205, 97\n  ], \n ... }\n}"],
  "untrusted": false
}
```


Example 3: Returned a missed (unexisting) transaction.


```json
$ curl http://127.0.0.1:18081/get_transactions -d '{"txs_hashes":
["d6e48158472848e6687173a91ae6eebfa3e1d778e65252ee99d7515d63090409"]}' -H 'Content-Type:
application/json'

{
  "credits": 0,
  "missed_tx": ["d6e48158472848e6687173a91ae6eebfa3e1d778e65252ee99d7515d63090409"],
  "status": "OK",
  "top_hash": "",
  "untrusted": false
}
```


### **/in_peers**

Limit number of Incoming peers.

Alias: _None_ .

Inputs:

- _in_peers_ - unsigned int; Max number of incoming peers

Outputs:

- _in_peers_ - unsigned int; Max number of incoming peers
- _status_ - string; General RPC error code. "OK" means everything looks good.
- _untrusted_ - boolean; States if the result is obtained using the bootstrap mode, and is therefore not trusted (`true`), or when the daemon is fully synced and thus handles the RPC locally (`false`)

Example:


```json
$ curl http://127.0.0.1:18081/in_peers -d '{"in_peers": 3232235535}' -H 'Content-Type:
application/json'
```

```
{
  "in_peers": 3232235535,
  "status": "OK",
  "untrusted": false
}
```

### **/is_key_image_spent**

Check if outputs have been spent using the key image associated with the output.

Alias: _None_ .

Inputs:

- _key_images_ - string list; List of key image hex strings to check.

Outputs:

- _credits_ - unsigned int; If payment for RPC is enabled, the number of credits available to the requesting client. Otherwise, 0.
- _spent_status_ - unsigned int list; List of statuses for each image checked. Statuses are follows: 0 = unspent, 1 = spent in blockchain, 2 = spent in transaction pool
- _status_ - string; General RPC error code. "OK" means everything looks good.
- _top_hash_ - string; If payment for RPC is enabled, the hash of the highest block in the chain. Otherwise, empty.
- _untrusted_ - boolean; States if the result is obtained using the bootstrap mode, and is therefore not trusted (`true`), or when the daemon is fully synced and thus handles the RPC locally (`false`)

Example:

```json
$ curl http://127.0.0.1:18081/is_key_image_spent -d '{"key_images":
["8d1bd8181bf7d857bdb281e0153d84cd55a3fcaa57c3e570f4a49f935850b5e3","7319134bfc50668251f5b
899c66b005805ee255c136f0e1cecbb0f3a912e09d4"]}' -H 'Content-Type: application/json'

{
  "credits": 0,
  "spent_status": [1,1],
  "status": "OK",
  "top_hash": "",
  "untrusted": false
}
```

### **/mining_status**

Get the mining status of the daemon.

Alias: _None_ .

Inputs: _None_ .

Outputs:

- _active_ - boolean; States if mining is enabled (`true`) or disabled (`false`).
- _address_ - string; Account address daemon is mining to. Empty if not mining.
- _bg_idle_threshold_ - int; Minimum average idle percentage over lookback interval.
- _bg_ignore_battery_ - boolean; If false, the device will stop mining when battery is low.
- _bg_min_idle_seconds_ - int; Minimum lookback interval in seconds for determining whether the device is idle or not.
- _bg_target_ - int; Maximum percentage cpu use by miner.
- _block_reward_ - int; Base block reward for the next block to be mined.

- _block_target_ - int; The target number of seconds between blocks.
- _difficulty_ - unsigned int; Network difficulty (analogous to the strength of the network)
- _difficulty_top64_ - unsigned int; Most-significant 64 bits of the 128-bit network difficulty.
- _is_background_mining_enabled_ - boolean; States if the mining is running in background (`true`) or foreground (`false`).
- _pow_algorithm_ - string; The name of the proof-of-work algorithm currently being used by the miner.
- _speed_ - unsigned int; Mining power in hashes per seconds.
- _status_ - string; General RPC error code. "OK" means everything looks good. Any other value means that something went wrong.
- _threads_count_ - unsigned int; Number of running mining threads.
- _untrusted_ - boolean; States if the result is obtained using the bootstrap mode, and is therefore not trusted (`true`), or when the daemon is fully synced and thus handles the RPC locally (`false`)
- _wide_difficulty_ - string; Network difficulty (analogous to the strength of the network) as a hexadecimal string representing a 128-bit number.

Example while mining:

```json
$ curl http://127.0.0.1:18081/mining_status -H 'Content-Type: application/json'

{
  "active": true,
  "address":
"47xu3gQpF569au9C2ajo5SSMrWji6xnoE5vhr94EzFRaKAGw6hEGFXYAwVADKuRpzsjiU1PtmaVgcjUJF89ghGPhU
XkndHc",
  "bg_idle_threshold": 0,
  "bg_ignore_battery": false,
  "bg_min_idle_seconds": 0,
  "bg_target": 0,
  "block_reward": 1181637918707,
  "block_target": 120,
  "difficulty": 239928394679,
  "difficulty_top64": 0,
  "is_background_mining_enabled": false,
  "pow_algorithm": "RandomX",
  "speed": 23,
  "status": "OK",
  "threads_count": 1,
  "untrusted": false,
  "wide_difficulty": "0x37dcd8c3b7"
}
```

Example while not mining:

```json
$ curl http://127.0.0.1:18081/mining_status -H 'Content-Type: application/json'

{
  "active": false,
  "address": "",
  "bg_idle_threshold": 0,
  "bg_ignore_battery": false,
  "bg_min_idle_seconds": 0,
  "bg_target": 0,
  "block_reward": 0,
  "block_target": 120,
  "difficulty": 239928394679,
  "difficulty_top64": 0,
  "is_background_mining_enabled": false,
  "pow_algorithm": "RandomX",
  "speed": 0,
  "status": "OK",
  "threads_count": 0,
  "untrusted": false,
```

```
  "wide_difficulty": "0x37dcd8c3b7"
}
```

### **/out_peers**

Limit number of Outgoing peers.

Alias: _None_ .

Inputs:

- _out_peers_ - unsigned int; Max number of outgoing peers

Outputs:

- _out_peers_ - unsigned int; Max number of outgoing peers
- _status_ - string; General RPC error code. "OK" means everything looks good.
- _untrusted_ - boolean; States if the result is obtained using the bootstrap mode, and is therefore not trusted (`true`), or when the daemon is fully synced and thus handles the RPC locally (`false`)

Example:

```json
$ curl http://127.0.0.1:18081/out_peers -d '{"out_peers": 3232235535}' -H 'Content-Type: application/json'

{
  "out_peers": 3232235535,
  "status": "OK",
  "untrusted": false
}
```

### **/pop_blocks**

Alias: _None_ .

Inputs:

- _nblocks_ - unsigned int;

Outputs:

- _height_ - unsigned int; New blockheight after popping blocks
- _status_ - string; General RPC error code. "OK" means everything looks good.
- _untrusted_ - boolean; States if the result is obtained using the bootstrap mode, and is therefore not trusted (`true`), or when the daemon is fully synced and thus handles the RPC locally (`false`)

Example:

```json
$ curl http://127.0.0.1:18081/pop_blocks -d '{"nblocks":6}' -H 'Content-Type: application/json''

{
  "height": 76482,
  "status": "OK",
  "untrusted": false
}
```

### **/save_bc**

Save the blockchain. The blockchain does not need saving and is always saved when modified, however it does a sync to flush the filesystem cache onto the disk for safety purposes against Operating System or Hardware crashes.

Alias: _None_ .

Inputs: _None_ .

Outputs:

- _status_ - string; General RPC error code. "OK" means everything looks good. Any other value means that something went wrong.
- _untrusted_ - boolean; States if the result is obtained using the bootstrap mode, and is therefore not trusted (`true`), or when the daemon is fully synced and thus handles the RPC locally (`false`)

Example:

```json
$ curl http://127.0.0.1:18081/save_bc -H 'Content-Type: application/json'

{
  "status": "OK",
  "untrusted": false
}
```

### **/send_raw_transaction**

Broadcast a raw transaction to the network.

Alias: _/sendrawtransaction_ .

Inputs:

- _tx_as_hex_ - string; Full transaction information as hexadecimal string.
- _do_not_relay_ - boolean; Stop relaying transaction to other nodes (default is `false`).

Outputs:

- _double_spend_ - boolean;  Transaction is a double spend (`true`) or not (`false`).
- _fee_too_low_ - boolean; Fee is too low (`true`) or OK (`false`).
- _invalid_input_ - boolean; Input is invalid (`true`) or valid (`false`).
- _invalid_output_ - boolean; Output is invalid (`true`) or valid (`false`).
- _low_mixin_ - boolean; Mixin count is too low (`true`) or OK (`false`).
- _not_rct_ - boolean; Transaction is a standard ring transaction (`true`) or a ring confidential transaction (`false`).
- _not_relayed_ - boolean; Transaction was not relayed (`true`) or relayed (`false`).
- _overspend_ - boolean; Transaction uses more money than available (`true`) or not (`false`).
- _reason_ - string; Additional information. Currently empty or "Not relayed" if transaction was accepted but not relayed.
- _status_ - string; General RPC error code. "OK" means everything looks good. Any other value means that something went wrong.
- _too_big_ - boolean; Transaction size is too big (`true`) or OK (`false`).
- _untrusted_ - boolean; States if the result is obtained using the bootstrap mode, and is therefore not trusted (`true`), or when the daemon is fully synced and thus handles the RPC locally (`false`)

Example (No return information included here.):

```json
```

```
$ curl http://127.0.0.1:18081/send_raw_transaction -d '{"tx_as_hex":"de6a3...",
"do_not_relay":false}' -H 'Content-Type: application/json'
```

### **/set_bootstrap_daemon**

Give immediate usability to wallets while syncing by proxying RPC requests.

Alias: _None_ .

Inputs:

- _address_ - string; host:port
- _username_ - string;
- _password_ - string;
- _proxy_ - string;

Outputs:

- _status_ - string; General RPC error code. "OK" means everything looks good.

Example:

Once set, the address will appear in `get_info` as `bootstrap_daemon_address`

```json
$ curl http://127.0.0.1:18081/set_bootstrap_daemon -d '{"address":
"http://getmonero.org:18081"}' -H 'Content-Type: application/json'

{
  "status": "OK"
}
```

### **/set_limit**

Set daemon bandwidth limits.

Alias: _None_ .

Inputs:

- _limit_down_ - signed int; Download limit in kBytes per second (-1 reset to default, 0 don't change the current limit)
- _limit_up_ - signed int; Upload limit in kBytes per second (-1 reset to default, 0 don't change the current limit)

Outputs:

- _limit_down_ - unsigned int; Download limit in kBytes per second
- _limit_up_ - unsigned int; Upload limit in kBytes per second
- _status_ - string; General RPC error code. "OK" means everything looks good.
- _untrusted_ - boolean; States if the result is obtained using the bootstrap mode, and is therefore not trusted (`true`), or when the daemon is fully synced and thus handles the RPC locally (`false`)

Example:

```json
$ curl http://127.0.0.1:18081/set_limit -d '{"limit_down": 1024}' -H 'Content-Type:
application/json'

{
  "limit_down": 1024,
  "limit_up": 128,
```

```
  "status": "OK"
  "untrusted": false
}
```

### **/set_log_categories**

Set the daemon log categories.
Categories are represented as a comma separated list of `<Category>:<level>` (similarly to syslog standard `<Facility>:<Severity-level>`), where:
- _Category_  is one of the following:
  - __ * - All facilities
  - _default_
  - _net_
  -  _net.http_
  -  _net.p2p_
  -  _logging_
  -  _net.throttle_
  -  _blockchain.db_
  -  _blockchain.db.lmdb_
  -  _bcutil_
  -  _checkpoints_
  -  _net.dns_
  -  _net.dl_
  -  _i18n_
  -  _perf_
  -  _stacktrace_
  -  _updates_
  -  _account_
  -  _cn_
  -  _difficulty_
  -  _hardfork_
  -  _miner_
  -  _blockchain_
  -  _txpool_
  -  _cn.block_queue_
  -  _net.cn_
  -  _daemon_
  -  _debugtools.deserialize_
  -  _debugtools.objectsizes_
  -  _device.ledger_
  -  _wallet.gen_multisig_
  -  _multisig_
  -  _bulletproofs_
  - _ringct_
  - _daemon.rpc_
  - _wallet.simplewallet_
  - _WalletAPI_
  - _wallet.ringdb_
  - _wallet.wallet2_
  - _wallet.rpc_
  - _tests.core_
- _Level_ is one of the following:
  - _FATAL_ - higher level
  - _ERROR_
  - _WARNING_
  - _INFO_
  - _DEBUG_
  - _TRACE_ - lower level
A level automatically includes higher level.
By default, categories are set to
`*:WARNING,net:FATAL,net.p2p:FATAL,net.cn:FATAL,global:INFO,verify:FATAL,stacktrace:INFO,logging:INFO,msgwriter:INFO`.
Setting the categories to "" prevents any logs from being outputted.

Alias: _None_ .

Inputs:

- _categories_ - string; Optional, daemon log categories to enable

Outputs:

- _categories_ - string; daemon log enabled categories
- _status_ - string; General RPC error code. "OK" means everything looks good. Any other value means that something went wrong.
- _untrusted_ - boolean; States if the result is obtained using the bootstrap mode, and is therefore not trusted (`true`), or when the daemon is fully synced and thus handles the RPC locally (`false`)

Example to set all facilities to Security Level `Info`:

```json
$ curl http://127.0.0.1:18081/set_log_categories -d '{"categories": "*:INFO"}' -H 'Content-Type: application/json'

{
  "categories": "*:INFO",
  "status": "OK",
  "untrusted": false
}
```

Example without input to set the default categories:

```json
$ curl http://127.0.0.1:18081/set_log_categories -H 'Content-Type: application/json'

{
  "categories":
"*:WARNING,net:FATAL,net.p2p:FATAL,net.cn:FATAL,global:INFO,verify:FATAL,stacktrace:INFO,logging:INFO,msgwriter:INFO",
  "status": "OK"
  "untrusted": false
}
```

### **/set_log_hash_rate**

Set the log hash rate display mode.

Alias: _None_ .

Inputs:

- _visible_ - boolean; States if hash rate logs should be visible (`true`) or hidden (`false`)

Outputs:

- _status_ - string; General RPC error code. "OK" means everything looks good. Any other value means that something went wrong.
- _untrusted_ - boolean; States if the result is obtained using the bootstrap mode, and is therefore not trusted (`true`), or when the daemon is fully synced and thus handles the RPC locally (`false`)

Example while mining:

```json
$ curl http://127.0.0.1:18081/set_log_hash_rate -d '{"visible":true}' -H 'Content-Type: application/json'

{
  "status": "OK"
```

```
  "untrusted": false
}
```

Error while not mining:

```json
$ curl http://127.0.0.1:18081/set_log_hash_rate -d '{"visible":true}' -H 'Content-Type:
application/json'

{
  "status": "NOT MINING",
  "untrusted": false
}
```

### **/set_log_level**

Set the daemon log level.
By default, log level is set to `0`.

Alias: _None_ .

Inputs:

- _level_ - integer; daemon log level to set from `0` (less verbose) to `4` (most verbose)

Outputs:

- _status_ - string; General RPC error code. "OK" means everything looks good. Any other value means that something went wrong.
- _untrusted_ - boolean; States if the result is obtained using the bootstrap mode, and is therefore not trusted (`true`), or when the daemon is fully synced and thus handles the RPC locally (`false`)

Example:

```json
$ curl http://127.0.0.1:18081/set_log_level -d '{"level":1}' -H 'Content-Type:
application/json'

{
  "status": "OK"
  "untrusted": false
}
```

### **/start_mining**

Start mining on the daemon.

Alias: _None_ .

Inputs:

- _do_background_mining_ - boolean; States if the mining should run in background (`true`) or foreground (`false`).
- _ignore_battery_ - boolean; States if battery state (on laptop) should be ignored (`true`) or not (`false`).
- _miner_address_ - string; Account address to mine to.
- _threads_count_ - unsigned int; Number of mining thread to run.

Outputs:

- _status_ - string; General RPC error code. "OK" means everything looks good. Any other value means that something went wrong.

- _untrusted_ - boolean; States if the result is obtained using the bootstrap mode, and is therefore not trusted (`true`), or when the daemon is fully synced and thus handles the RPC locally (`false`)

Example:

```json
$ curl http://127.0.0.1:18081/start_mining -d
'{"do_background_mining":false,"ignore_battery":true,"miner_address":"47xu3gQpF569au9C2ajo
5SSMrWji6xnoE5vhr94EzFRaKAGw6hEGFXYAwVADKuRpzsjiU1PtmaVgcjUJF89ghGPhUXkndHc","threads_coun
t":1}' -H 'Content-Type: application/json'

{
  "status": "OK",
  "untrusted": false
}
```

### **/start_save_graph**

Obsolete. Conserved here for reference.

Alias: _None_ .

Inputs: _None_ .

Outputs:

- _status_ - string; General RPC error code. "OK" means everything looks good.

Example:

```json
$ curl http://127.0.0.1:18081/start_save_graph -H 'Content-Type: application/json'

{
  "status": "OK"
}
```

### **/stop_daemon**

Send a command to the daemon to safely disconnect and shut down.

Alias: _None_ .

Inputs: _None_ .

Outputs:

- _status_ - string; General RPC error code. "OK" means everything looks good.

Example:

```json
$ curl http://127.0.0.1:18081/stop_daemon -H 'Content-Type: application/json'

{
  "status": "OK"
}
```

### **/stop_mining**

Stop mining on the daemon.

Alias: _None_ .

Inputs: _None_ .

Outputs:

- _status_ - string; General RPC error code. "OK" means everything looks good. Any other value means that something went wrong.
- _untrusted_ - boolean; States if the result is obtained using the bootstrap mode, and is therefore not trusted (`true`), or when the daemon is fully synced and thus handles the RPC locally (`false`)

Example:

```json
$ curl http://127.0.0.1:18081/stop_mining -H 'Content-Type: application/json'

{
  "status": "OK",
  "untrusted": false
}
```

### **/stop_save_graph**

Obsolete. Conserved here for reference.

Alias: _None_ .

Inputs: _None_ .

Outputs:

- _status_ - string; General RPC error code. "OK" means everything looks good.

Example:

```json
$ curl http://127.0.0.1:18081/stop_save_graph -H 'Content-Type: application/json'

{
  "status": "OK"
}
```

### **/update**

Update daemon.

Alias: _None_ .

Inputs:

- _command_ - String; command to use, either `check` or `download`
- _path_ - String; Optional, path where to download the update.

Outputs:

- _auto_uri_ - string;
- _hash_ - string;
- _path_ - String; path to download the update
- _status_ - string; General RPC error code. "OK" means everything looks good.

- _untrusted_ - boolean; States if the result is obtained using the bootstrap mode, and is therefore not trusted (`true`), or when the daemon is fully synced and thus handles the RPC locally (`false`)
- _update_ - boolean; States if an update is available to download (`true`) or not (`false`)
- _user_uri_ - string;
- _version_ - string; Version available for download.

Example:

```json
$ curl http://127.0.0.1:18081/update -d '{"command":"check"}' -H 'Content-Type: application/json'

{
  "auto_uri": "",
  "hash": "",
  "path": "",
  "status": "OK",
  "untrusted": false,
  "update": false,
  "user_uri": "",
  "version": ""
}
```

## Sources:

Reworked from [monero-site](https://github.com/monero-project/monero-site/commit/297e34008744e5af1bd1f0cd2ca1edda802fe895)