YTA RPC API手册

YTA规定了节点旳JSON RPC API开发接口，根据实现插件的不同，这些API分别归入 以下分组：

- CHAIN：由chain_api_plugin实现，主要提供区块链数据的访问功能
- HISTORY：由history_api_plugin实现，主要提供区块链历史交易的访问功能
- NET：由net_api_plugin实现，主要提供P2P网络管理功能
- PRODUCER：由producer_api_plugin实现，主要提供出块管理功能
- DBSIZE：由db_size_api_plugin实现，主要提供数据库相关功能

YTA RPC API的当前版本为1.0.0，官方节点软件nodeos默认在8888端口监听RPC API访问请求。

CHAIN

CHAIN分组中的RPC API，为开发者提供访问区块链的能力，主要包含以下API：

- get_info：区块链概要信息查询
- get_block：区块数据查询
- get_block_header_state：区块头查询
- get_account：账号信息查询
- get_abi：合约abi查询
- get_code：合约代码查询
- get_raw_code_and_abi：合约abi和代码原始数据查询
- get_table_rows：多索引表查询
- get_table_by_scope：分作用域的多索引表查询
- get_currency_balance：代币余额查询
- abi_json_to_bin：合约动作调用序列化
- abi_bin_to_json：合约动作调用反序列化
- get_required_keys：交易签名所需公钥查询
- get_currency_stats：代币发行信息查询
- get_producers：出块账号查询
- get_producer_schedule：块生产者顺序查询
- push_block：区块提交
- push_transaction：交易提交
- push_transactions：交易组提交

HISTORY

HISTORY分组中的RPC API，为开发者提供访问历史交易的能力，主要包含以下API：

- get_actions：查询历史动作
- get_transactions：查询历史交易
- get_key_accounts：查询与指定公钥关联的账号

- get_controlled_accounts：查询指定账号的受控子账号

NET

NET分组中的RPC API，为开发者提供管理P2P网络的能力，主要包含以下API：

- connect：连接指定的P2P节点
- disconnect：断开与指定P2P节点旳连接
- connections：查询P2P网络的连接情况
- status：查询P2P网络的状态

PRODUCER

PRODUCER分组中的RPC API，为开发者提供管理出块模块的能力，主要包含以下API：

- pause：暂停出块
- resume：恢复出块
- paused：查询当前出块状态
- get_runtime_options：查询出块运行参数
- update_runtime_options：更新出块运行参数
- get_greylist：查询出块灰名单
- add_greylist_accounts：将指定账号添加到出块灰名单
- remove_greylist_accounts：从出块灰名单移除指定账号
- get_whitelist_blacklist：查询出块白名单和黑名单
- set_whitelist_blacklist：设置出块白名单和黑名单

DBSIZE

DBSIZE分组中的RPC API，为开发者提供数据库相关信息，主要包含以下API：

- get：获取数据信息

CHAIN

get_info

chain/get_info调用返回区块链总体信息。

调用参数

无

返回值

get_info调用的返回结果是一个JSON对象，包含当前所连接EOS链的总体信息，成员如下：

```
server_version：节点版本
head_block_num：链头区块序号
lase_irreversible_block_num：最后不可逆区块号
head_block_id：链头区块ID
```

```
head_block_time：链头区块生成时间
head_block_producer：链头区块出块账号
```

示例代码

调用请求：

```
~$ curl -X POST --url http://127.0.0.1:8888/v1/chain/get_info
```

返回结果：


```
{
    "server_version": "d9ad8eec",
    "head_block_num": 8592,
    "last_irreversible_block_num": 8591,
    "head_block_id": "00002190e805475db152be7d3f4f1a075efaed42827cd551b0e23c7feabbedac",
    "head_block_time": "2018-04-27T17:40:34",
    "head_block_producer": "eosio"
}
```


get_block

chain/get_block调用返回指定区块的详细数据。

注意，nodeos时需要启用chain_api_plugin插件。

调用参数

调用参数为一个JSON对象，用来指定要提取数据的区块，其成员如下：

```
block_num_or_id：字符串，要提取数据的区块序号或ID
```

返回值

get_block调用返回描述指定区块数据的JSON对象，其成员如下：

```
previous：前一个区块的ID，字符串
timestamp：区块时间戳，时间字符串
transaction_mroot：交易默克尔树根
action_mroot：action的默克尔树根
block_mroot：区块默克尔树根
producer：出块账号，字符串
schedule_version：调度器版本，数值
new_producers：新出现的出块账号集合，空或数组
producer_signature：出块账号的签名，字符串
regions：分区集合，数组
input_transactions：区块中包含的交易集合，数组
id：区块ID，字符串
block_num：区块序号，整数
ref_block_prefix：参考块前缀，整数
```

示例代码

调用请求：

```
~$ curl -X POST --url http://127.0.0.1:8888/v1/chain/get_block -d '{
  "block_num_or_id": "000000056d75b0581b4fbb96affa36669a37173d21f46f8cb974f760e94bbe14"
}'
```

返回结果：


```
{
    "previous": "00000004471d48fe40706e73ce27f9cf7bac1704ae55279c7a58c0173718a711",
    "timestamp": "2018-04-18T16:24:23.500",
    "transaction_mroot": "e366c0cc3519bb0f2ddaec20928fa4d6aae546194bb1c4205c67be429147ed4a",
    "action_mroot": "77e5e91b594ab4ebc44ebc8c7ecdc9d26409c5a07452d3b20a4840562fdeb658",
    "block_mroot": "4ef85b0d212f3fffabdd65680d32dd7dded3461d9df226a6e3dc232e42978f8b",
    "producer": "eosio",
    "schedule_version": 0,
    "new_producers": null,
    "producer_signature":
"YTAJzEdFDsueKCerL7a6AdxMxiT851cEiugFB7ux1PAGn5eMmco8j32NsaKupxibheQGVFEqyEdjMub67VZjKmsLzuNxxKtUA",
    "regions": [{
        "region": 0,
        "cycles_summary": [
            [{
                "read_locks": [],
                "write_locks": [],
                "transactions": [{
                    "status": "executed",
                    "kcpu_usage": 2,
                    "net_usage_words": 38,
                    "id": "9880c128683e24845ccd282ebe026bd522f7fa9c6278d885f6ed35164c680669"
                }]
            }]
        ]
    }],
    "input_transactions": [],
    "id": "000000056d75b0581b4fbb96affa36669a37173d21f46f8cb974f760e94bbe14",
    "block_num": 5,
    "ref_block_prefix": 2528857883
}
```


get_block_header_state

chain/get_block_header_state调用返回指定区块的头信息。

注意，nodeos时需要启用chain_api_plugin插件。

调用参数

JSON对象，用来指定要获取块头信息的区块，成员如下：

```
block_num_or_id：区块序号或ID，字符串
```

返回值

get_block_header_state调用返回一个描述区块头信息的JSON对象。

示例代码

调用请求：

```
~$ curl -X POST --url http://127.0.0.1:8888/v1/chain/get_block_header_state -d '{
  "block_num_or_id": "4"
}'
```

返回结果：

```
{
  ...
}
```

get_account

chain/get_account调用返回指定账号的描述信息。

注意，nodeos时需要启用chain_api_plugin插件。

调用参数

JSON对象，用来指定要读取信息的账号，其成员如下：

```
account_name：账号名称，字符串
```

返回值

get_account调用的返回值是描述账号信息的JSON对象，其成员如下：

- account_name：账号名称
- permissions: 账号权限集合，权限对象数组，每个权限对象的成员如下：
  - perm_name：权限名称
  - parent：父权限名称
  - required_auth：授权条件，JSON对象，成员如下：
    - threshold：阈值，整数
    - keys：公钥信息集合，数组，每个公钥信息对象成员如下：
      - key：公钥，字符串
      - weight：权重
      - accounts：账号数组

示例代码

调用请求：

```
~$ curl -X POST --url http://127.0.0.1:8888/v1/chain/get_account -d '{
  "account_name": "eosio"
}'
```

返回结果：

```
{
    "account_name": "eosio",
    "permissions": [{
        "perm_name": "active",
        "parent": "owner",
        "required_auth": {
            "threshold": 1,
            "keys": [{
                "key": "YTA6MRyAjQq8ud7hVNYcfnVPJqcVpscN5So8BhtHuGYqET5GDW5CV",
                "weight": 1
            }],
            "accounts": []
        }
    }, {
        "perm_name": "owner",
        "parent": "",
        "required_auth": {
            "threshold": 1,
            "keys": [{
                "key": "YTA6MRyAjQq8ud7hVNYcfnVPJqcVpscN5So8BhtHuGYqET5GDW5CV",
                "weight": 1
            }],
            "accounts": []
        }
    }]
}
```

get_abi

chain/get_abi调用返回指定账号所托管合约的abi描述信息。

注意，nodeos时需要启用chain_api_plugin插件。

调用参数

JSON对象，用来指定要查询的合约托管账户，其成员如下：

```
account_name：账户名称，字符串
```

返回值

get_abi调用返回指定账号上所部署的合约的abi描述对象。 示例代码

调用请求：

```
~$ curl -X POST --url http://127.0.0.1:8888/v1/chain/get_abi -d '{
  "account_name": "eosio"
```

```
}'
```

返回结果：

```
{
  ...
}
```

get_code

chain/get_code调用返回指定账号所托管合约的代码信息。

注意，nodeos时需要启用chain_api_plugin插件。

调用参数

JSON对象，用来声明要查询代码的账号，其成员如下：

```
account_name：账号名称，字符串
```

返回值

get_code调用返回指定账号上部署的合约的代码描述对象，其成员如下：

- name：账号名称，字符串
- code_hash：当前部署代码的哈希值，字符串
- wast：当前部署代码的等价文本格式，字符串
- abi：当前部署代码的abi描述对象，其成员如下：
  - types：abi中定义的新类型，数组
  - structs：abi中定义的新结构，数组
  - actions：abi中定义的合约动作，数组
  - tables：abi中定义的数据表，数组

示例代码

调用请求：

注意该接口修改了源码，不支持返回wast数据了。因此在请求该接口的时候，要使用"code_as_wasm"参数。

```
~$ curl -X POST --url http://127.0.0.1:8888/v1/chain/get_code -d '{
  "account_name" : "currency",
  "code_as_wasm" : true
}'
```

返回结果：

```
{
  "name":"currency",
```

```
  "code_hash":"a1c8c84b4700c09c8edb83522237439e33cf011a4d7ace51075998bd002e04c9",
  "wast":"(module\n  (type $0 (func (param i64 i64 i32) (result i32)))\n ...truncated",
  "abi": {
  "types": [{
      "new_type_name": "account_name",
      "type": "name"
    }
  ],
  "structs": [{
      "name": "transfer",
      "base": "",
      "fields": [
        {"name":"from", "type":"account_name"},
        {"name":"to", "type":"account_name"},
        {"name":"quantity", "type":"uint64"}
      ]
    },{
      "name": "account",
      "base": "",
      "fields": [
        {"name":"key", "type":"name"},
        {"name":"balance", "type":"uint64"}
      ]
    }
  ],
  "actions": [{
      "name": "transfer",
      "type": "transfer"
    }
  ],
  "tables": [{
      "name": "account",
      "type": "account",
      "index_type": "i64",
      "key_names" : ["key"],
      "key_types" : ["name"]
    }
  ]
}
```


get_raw_code_and_abi

chain/get_raw_code_and_abi调用返回指定账号所部署合约的代码和合约abi的原始格式数据。

调用参数

JSON对象，用来指定一个账号，其成员如下：

```
account_name：账号名称，字符串
```

返回值

get_raw_code_and_abi调用的返回值为合约代码与abi的原始描述对象。

示例代码

调用请求：

```
~$ curl -X POST --url http://127.0.0.1:8888/v1/chain/get_raw_code_and_abi -d '{
  "account_name": "eosio"
}'
```

返回结果：

```
{
  ...
}
```

get_table_rows

chain/get_table_rows调用指定多索引数据表的数据行。

注意，nodeos时需要启用chain_api_plugin插件。

调用参数

JSON对象，指定数据表查询条件，其成员如下：

- scope：数据表作用域，字符串
- code：合约的托管账号名，字符串
- table：数据表名称，字符串
- table_key：键名称，字符串，可选
- json：是否返回JSON格式的结果，布尔型，默认值：true
- lower_bound：结果数据应当满足的下界值，字符串，可选
- upper_bound：结果数据应当满足的上界值，字符串，可选
- limit：返回结果数量上限，整数，默认值：10
- index_position：要使用的索引序号，例如，主键索引为1，次级索引为2，字符串，默认值：1
- key_type：索引键类型，例如uint64_t或name，字符串
- encode_type：编码类型，字符串，默认值：dec

返回值

get_table_rows调用的返回值为满足查询条件的数据行信息。

示例代码

调用请求：

```
~$ curl -X POST --url http://127.0.0.1:8888/v1/chain/get_table_rows  -d '{
  "code":"currency",
  "scope": "inita",
  "table": "account",
  "json": true
}'
```

返回结果：

```
{
  "rows": [
    {
      "account": "account",
      "balance": 1000
    }
  ],
  "more": false
}
```

get_table_by_scope

chain/get_table_by_scope调用返回多索引数据表中满足指定查询条件的数据行。

调用参数

JSON对象，指定数据表查询条件，其成员如下：

- code：合约的托管账号名，字符串
- table：数据表名称，字符串
- lower_bound：结果数据应当满足的下界值，字符串，可选
- upper_bound：结果数据应当满足的上界值，字符串，可选
- limit：返回结果数量上限，整数，默认值：10

返回值

get_table_by_scope调用的返回值为满足查询条件的数据行信息。

示例代码

调用请求：

```
~$ curl -X POST --url http://127.0.0.1:8888/v1/chain/get_table_by_scope -d '{
  "code": "eosio.token",
  "table": "stats"
}'
```

返回结果：

```
{
  ...
}
```

get_currency_balance

chain/get_currency_balance调用返回指定账户的代币余额信息。

注意，nodeos时需要启用chain_api_plugin插件。

调用参数

JSON对象，用来声明查询参数，其成员如下：

- code：代币合约托管账户名称，字符串
- account：要查询账户的名称，字符串
- symbol：要查询的代码符号，字符串

返回值

get_currency_balance调用返回指定账号所持有的代币余额。

示例代码

调用请求：

```
~$ curl -X POST --url http://127.0.0.1:8888/v1/chain/get_currency_balance -d '{
  "code":"eosio.token",
  "account": "eosio",
  "symbol": "YTA"
}'
```

返回结果：

```
{
  ...
}
```

abi_json_to_bin

chain/abi_json_to_bin调用将合约动作调用序列化为16进制字符串，以便应用于 push_action调用。

注意，nodeos时需要启用chain_api_plugin插件。

调用参数

JSON对象，用来声明要进行序列化的合约动作调用，其成员如下：

- code：合约托管账号名称，字符串
- action：要调用的合约动作名称，字符串
- args：合约动作参数，JSON对象

返回值

abi_json_to_bin调用的返回结果为序列化字符串。

示例代码

调用请求：

```
~$ curl -X POST --url http://127.0.0.1:8888/v1/chain/abi_json_to_bin -d '{
  "code": "eosio.token",
  "action":"transfer",
```

```
  "args": {"from":"eosio","to":"usernameabcd","quantity":"8.0000 YTA","memo":"memo"}
}'
```

返回结果：

```
{
  "binargs": "0000000000ea305590d0314a9a7915d6803801000000000004595441000000000046d656d6f"
}
```

abi_bin_to_json

chain/abi_bin_to_json调用将16进制码流反序列化为合约的动作调用。

注意，nodeos时需要启用chain_api_plugin插件。

调用参数

JSON对象，用来声明要进行序列化的合约动作调用，其成员如下：

- code：合约托管账号名称，字符串
- action：要调用的合约动作名称，字符串
- args：合约动作参数，16进制码流，字符串

返回结果

abi_bin_to_json调用的返回结果为反序列化后的合约动作调用对象。

示例代码

调用请求：

```
~$ curl -X POST --url http://127.0.0.1:8888/v1/chain/abi_bin_to_json -d '{
  "code":"eosio.token",
  "action":"transfer",
  "binargs": "0000000000ea305590d0314a9a7915d6803801000000000004595441000000000046d656d6f"
}'
```

返回结果：

```
{
  "args": {
    "from": "eosio",
    "to": "usernameabcd",
    "quantity": "8.0000 YTA",
    "memo": "memo"
  }
}
```

get_required_keys

chain/get_required_keys调用返回签名一个交易时需要的公钥清单。

注意，nodeos时需要启用chain_api_plugin插件。

调用参数

JSON对象，用来声明要签名的交易和可用的公钥集合，其成员如下：

- transaction：交易数据，JSON对象
- available_keys：可用公钥，数组

返回值

get_required_keys返回一组必须的用于签名的公钥。

示例代码

调用请求：

```
~$ curl -X POST --url http://127.0.0.1:8888/v1/chain/get_required_keys -d '{
  "transaction": {
    "ref_block_num": "100",
    "ref_block_prefix": "137469861",
    "expiration": "2017-09-25T06:28:49",
    "scope": ["initb", "initc"],
    "actions": [{
      "code": "eosio.token",
      "type": "transfer",
      "recipients": ["initb", "initc"],
      "authorization": [{
        "account": "initb",
        "permission": "active"
      }],
      "data": "000000000041934b000000008041934be803000000000000"
    }],
    "signatures": [],
    "authorizations": []
  },
  "available_keys": [
    "YTA4toFS3YXEQCkuuw1aqDLrtHim86Gz9u3hBdcBw5KNPZcursVHq",
    "YTA7d9A3uLe6As66jzN8j44TXJUqJSK3bFjjEEqR4oTvNAB3iM9SA",
    "YTA6MRyAjQq8ud7hVNYcfnVPJqcVpscN5So8BhtHuGYqET5GDW5CV"]
}'
```

返回结果：

```
{
  "required_keys": [
    "YTA6MRyAjQq8ud7hVNYcfnVPJqcVpscN5So8BhtHuGYqET5GDW5CV"
  ]
}
```

get_currency_stats

chain/get_currency_stats调用返回指定代币的总体信息。

注意，nodeos时需要启用chain_api_plugin插件。

调用参数

JSON对象，声明代币合约托管账号和代币信息，成员如下：

- code：代币合约的部署账号，字符串
- symbol：要查询的代币符号，字符串

返回值

get_currency_stats调用返回指定种类代币的总体发行信息。

示例代码

调用请求：

```
~$ curl -X POST --url http://127.0.0.1:8888/v1/chain/get_currency_stats -d '{
  "code":"eosio.token",
  "symbol": "YTA"
}'
```

返回结果：

```
{
  ...
}
```

get_producer_schedule

chain/get_producer_schedule 调用返回当前块生产者的顺序表。

注意，nodeos时需要启用chain_api_plugin插件。

调用参数

JSON对象，声明查询条件，其成员如下：

- json：是否返回JSON对象，布尔值，默认：true

返回值

get_producer_schedule 调用返回当前块生产者的顺序表。

示例代码

调用请求：

```
~$ curl -X POST --url http://127.0.0.1:8888/v1/chain/get_producer_schedule
```

返回结果：

```
{
  ...
}
```

get_producers

chain/get_producers调用返回当前生效的出块账号信息。

注意，nodeos时需要启用chain_api_plugin插件。

调用参数

JSON对象，声明查询条件，其成员如下：

- limit：返回结果的数量上限，字符串，可选
- lower_bound：结果应当满足的下界，字符串，可选
- json：是否返回JSON对象，布尔值，默认：true

返回值

get_producers调用返回当前生效的出块账号信息。

示例代码

调用请求：

```
~$ curl -X POST --url http://127.0.0.1:8888/v1/chain/get_producers
```

返回结果：

```
{
  ...
}
```

push_block

chain/push_block调用将指定的区块数据提交到链上。

注意，节点软件nodeos需要启用chain_api_plugin插件。

调用参数

JSON对象，声明要提交的区块数据，成员如下：

```
timestamp：区块时间戳，时间字符串
producer：出块账号，字符串
confirmed：生产者确认数
previous：前一个区块的ID，字符串
transaction_mroot: 交易默克尔树根
action_mroot: action默克尔树根
```

```
block_mroot:区块默克尔树根
version
new_producers：新出现的出块账号集合，空或数组
header_extensions：区块头扩展字段
producer_signature：出块账号的签名，字符串
transactions：区块中包含的交易集合，数组
block_extensions: 区块扩展字段
```

返回值

示例代码

调用请求：

```
~$ curl -X POST --url http://127.0.0.1:8888/v1/chain/push_block -d '{
  ...
}'
```

返回结果：

```
{
  ...
}
```

push_transaction

chain/push_transaction调用将指定的交易提交到链上。

注意，节点软件nodeos需要启用chain_api_plugin插件。

调用参数

JSON对象，声明交易、签名等信息，成员如下：

- signatures：签名数组
- compression：是否压缩格式，布尔类型，默认值：false
- packed_context_free_data：上下文无关的数据
- packed_tx：序列化的交易数据

返回值

push_transaction调用的返回结果包含交易ID。

示例代码

调用请求：

```
~$ curl -X POST --url http://127.0.0.1:8888/v1/chain/push_transaction -d '{
  ...
}'
```

返回结果：

```
{
    'transaction_id' = "1..."
}
```

push_transactions

chain/push_transactions调用将一组交易提交到链上。

注意，节点软件nodeos需要启用chain_api_plugin插件。

调用参数

JSON对象，指定一组要提交的交易，成员如下：

- body

返回值

push_transactions调用的返回值包含交易ID

示例代码

调用请求：

```
~$ curl -X POST --url http://127.0.0.1:8888/v1/chain/push_transactions -d '{
  ...
}'
```

返回结果：

```
{
    'transaction_id' = "1..."
}
```

HISTORY

get_actions

history/get_actions调用返回指定合约上执行过的动作。

注意，nodeos时需要启用history_api_plugin插件，并且设置filter-on选项。

调用参数

JSON对象，指定查询参数，其成员如下：

- pos：位置，整数

- offset：偏移量，整数
- account_name：合约托管账号，字符串

返回值

get_actions调用的返回值为指定账号的托管合约上发生过的历史动作调用。

示例代码

调用请求：

```
~$ curl -X POST --url http://127.0.0.1:8888/v1/history/get_actions -d '{
  "pos": -1,
  "offset": -20,
  "account_name": "eosio"
}'
```

返回结果：

```
{
  ...
}
```

get_transaction

history/get_transaction调用返回指定的交易数据。

注意，nodeos时需要启用history_api_plugin插件，并且设置filter-on选项。

调用参数

JSON对象，声明要查询的交易，其成员如下：

- id：交易ID，字符串

返回值

get_transaction调用的返回值为查询到的交易描述JSON对象。

示例代码

调用请求：

```
~$ curl -X POST --url http://127.0.0.1:8888/v1/history/get_transaction -d '{
  "id": "...."
}'
```

返回结果：

```
{
  ...
```

```
}
```

get_key_accounts

history/get_key_accounts调用返回指定公钥所关联的账号。

注意，nodeos时需要启用history_api_plugin插件，并且设置filter-on选项。

调用参数

JSON对象，声明要查询的公钥，其成员为：

```
public_key：公钥，字符串
```

返回值

get_key_accounts调用的返回值为一组关联指定公钥的账号。

示例代码

调用请求：

```
~$ curl -X POST --url http://127.0.0.1:8888/v1/history/get_key_accounts -d '{
  "public_key": "..."
}'
```

返回结果：

```
{
  ...
}
```

get_controlled_accounts

history/get_controlled_accounts调用返回指定账号的受控账号。

注意，nodeos时需要启用history_api_plugin插件，并且设置filter-on选项。

调用参数

JSON对象，声明主控账号，其成员为：

- controlling_account：主控账号名称，字符串

返回值

get_controlled_accounts调用返回一组受控账号。

示例代码

调用请求：

```
~$ curl -X POST --url http://127.0.0.1:8888/v1/history/get_controlled_accounts -d '{
  "controlling_account": "eosio"
}'
```

返回结果：

```
{
  ...
}
```

NET

connect

net/connect调用的作用是让当前节点连接指定的eos节点。

注意，nodeos时需要启用net_api_plugin插件。

调用参数

connect调用的参数为要连接的eos节点旳地址，字符串。

返回值

示例代码

调用请求：

```
~$ curl -X POST --url http://127.0.0.1:8888/v1/net/connect -d 1.2.3.4:9876'
```

返回结果：

disconnect

net/disconnect调用可以断开与指定eos节点的连接。

注意，nodeos时需要启用net_api_plugin插件。

调用参数

disconnect的参数为要断开连接的节点地址。

返回值

示例代码

调用请求：

```
~$ curl -X POST --url http://127.0.0.1:8888/v1/net/disconnect -d 'a.b.c.d:9876'
```

connections

net/connections调用返回当前节点旳P2P连接信息。

注意，nodeos时需要启用net_api_plugin插件。

调用参数

无

返回值

connections调用返回当前节点与其他节点旳连接信息。

示例代码

调用请求：

```
~$ curl -X POST --url http://127.0.0.1:8888/v1/net/connections
```

返回结果：

```
{
  ...
}
```

status

net/status调用返回当前节点旳P2P网络状态。

注意，节点软件nodeos需要启用net_api_plugin插件。

调用参数

无

返回值

status调用的返回值为一个描述网络状态的JSON对象。

示例代码

调用请求：

```
~$ curl -X POST --url http://127.0.0.1:8888/v1/net/status
```

返回结果：

```
{
...
}
```

PRODUCER

pause

producer/pause调用停用当前出块器。

注意，nodeos时需要启用producer_api_plugin插件。

调用参数

无

返回值

示例代码

调用请求：

```
~$ curl -X POST --url http://127.0.0.1:8888/v1/producer/pause
```

返回结果：

```
{
  ...
}
```

resume

producer/resume调用重新启用当前出块器。

注意，节点软件nodeos需要启用producer_api_plugin插件。

调用参数

无

返回结果

示例代码

调用请求：

```
~$ curl -X POST --url http://127.0.0.1:8888/v1/producer/resume
```

返回结果：

```
{
  ...
}
```

paused

producer/paused调用返回当前出块器是否暂停。

注意，nodeos时需要启用producer_api_plugin插件。

调用参数

无

返回结果

示例代码

调用请求：

```
~$ curl -X POST --url http://127.0.0.1:8888/v1/producer/paused
```

返回结果：

```
{
  ...
}
```

get_runtime_options

producer/get_runtime_options调用返回当前出块器的运行参数。

注意，nodeos时需要启用producer_api_plugin插件。

调用参数

无

返回值

get_runtime_options调用的返回结果为描述出块运行参数的JSON对象。

示例代码

调用请求：

```
~$ curl -X POST --url http://127.0.0.1:8888/v1/producer/get_runtime_options
```

返回结果：

```
{
  ...
}
```

update_runtime_options

producer/update_runtime_options调用可以修改出块运行参数。

注意，节点软件nodeos需要启用producer_api_plugin插件。

调用参数

JSON对象，描述目标运行参数，其成员为：

- max_transaction_time：交易时间上限，整数
- max_irreversible_block_age：不可逆块龄上限，整数
- produce_time_offset_us：出块时间偏移量，整数，单位：微秒
- last_block_time_offset_us：最新块时间偏移量，整数，单位：微秒
- incoming_defer_ratio：滞后率，整数
- subjective_cpu_leeway_us：整数，单位：微秒

返回值

示例代码

调用请求：

```
~$ curl -X POST --url http://127.0.0.1:8888/v1/producer/update_runtime_options -d '{
  "max_transaction_time": 0,
  "max_irreversible_block_age":0 ,
  "produce_time_offset_us": 0,
  "last_block_time_offset_us": 0,
  "incoming_defer_ratio": 0,
  "subjective_cpu_leeway_us": 0
}'
```

返回结果：

```
{
  ...
}
```

get_greylist

producer/get_greylist调用返回出块灰名单。

注意，nodeos时需要启用producer_api_plugin插件。

调用参数

无

返回值

get_greylist调用返回当前启用的出块灰名单。

示例代码

调用请求：

```
~$ curl -X POST --url http://127.0.0.1:8888/v1/producer/get_greylist
```

返回结果：

```
{
  "accounts":["a11111111111","aaa","bbb"]
}
```

add_greylist_accounts

producer/add_greylist_accounts调用向出块灰名单中添加指定的账号。

注意，nodeos时需要启用producer_api_plugin插件。

调用参数

JSON对象，声明要添加到灰名单中的账号，其成员为：

- accounts：字符串数组，每个成员指定一个账号

返回值

示例代码

调用请求：

```
~$ curl -X POST --url http://127.0.0.1:8888/v1/producer/add_greylist_accounts -d '{
  "accounts":["aaa","bbb"]
}'
```

返回结果：

```
{
  ...
}
```

remove_greylist_accounts

producer/remove_greylist_accounts调用从出块灰名单中删除指定的账号。

注意，节点软件nodeos需要启用producer_api_plugin插件。

调用参数

JSON对象，声明要添加到灰名单中的账号，其成员为：

- accounts：字符串数组，每个成员指定一个账号

返回值

示例代码

调用请求：

```
~$ curl -X POST --url http://127.0.0.1:8888/v1/producer/remove_greylist_accounts -d '{
  "accounts": ["aaa","bbb"]
}'
```

返回结果：

```
{
  ...
}
```

get_whitelist_blacklist

producer/get_whitelist_blacklist调用返回当前应用的出块白名单和黑名单。

注意，nodeos时需要启用producer_api_plugin插件。

调用参数

无

返回值

get_whitelist_blacklist调用返回一个JSON对象，包含出块白名单账号和黑名单账号。

示例代码

调用请求：

```
~$ curl -X POST --url http://127.0.0.1:8888/v1/producer/get_whitelist_blacklist
```

返回结果：

```
{
  ...
}
```

set_whitelist_blacklist

producer/set_whitelist_blacklist调用设置出块白名单和黑名单。

注意，节点软件nodeos需要启用producer_api_plugin插件。

调用参数

返回值

示例代码

调用请求：

```
~$ curl -X POST --url http://127.0.0.1:8888/v1/producer/set_whitelist_blacklist
```

返回结果：

DBSIZE

get

db_size/get调用返回数据大小。

注意，nodeos时需要启用db_size_api_plugin插件。

调用参数

无

返回值

示例代码

调用请求：

```
~$ curl -X POST --url http://127.0.0.1:8888/v1/db_size/get
```

返回结果：

```
{
  ...
}
```