DERO Atlantis RPC API V2.0

Dero Project - dero.io

September 4, 2018

Contents

Chapter 1

Introduction

This document describes the RPC API for the Dero daemon and wallet which are implemented according to the JSON RPC 2.0 standard.
We will give a description of the available RPC methods with their parameters and results and provide code examples for calling the methods and the data returned.

Dero is the first crypto project to combine a Proof of Work blockchain with a DAG (Directed Acyclic Graph) block structure and wholly anonymous transactions. The fully distributed ledger processes transactions with a twelve-second average block time and is secure against majority hashrate attacks.

Dero will be the first CryptoNote blockchain to have smart contracts on its native chain without any extra layers or secondary blockchains.

For more information visit http://www.dero.io

1.1 Data types

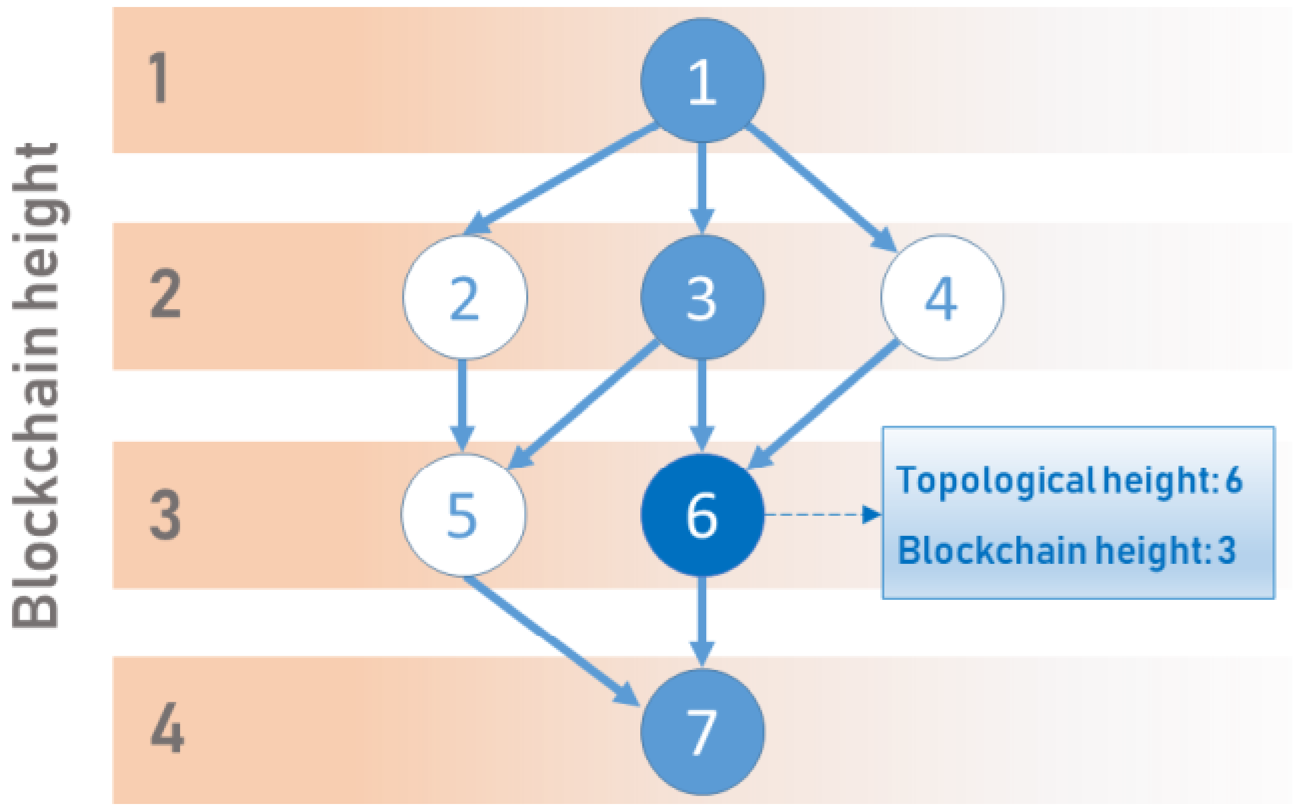
Dero is written in Go, so we give the data types of the parameters and results in Go format. It is pretty straightforward to convert them to other languages.

Amounts in Dero have a resolution of 1^{12} decimals and are handled as unsigned 64 bit integers.

```
e.g. 1.5 Dero is encoded as 1500000000000
```

As Dero combines DAG and Blockchain, it uses different height information than traditional blockchains. Dero blocks have an additional height value, called the topological height.

The topological height is unique for each block, while at each blockchain height there can be multiple blocks associated. Each blockchain height contains at least a main block and optional side blocks.



1.2 Code examples

The examples provided for each method are written in Python - using the 'request' package to build the HTTP request and perform the JSON encoding.

```
#construct payload for HTTP request
payload = {'jsonrpc': '2.0', 'id':'1', 'method': 'getblockcount'}

#send request
r = requests.post('http://127.0.0.1:20206/json_rpc', json=payload, headers={'
    Connection':'close'})

#read response object
if (r.status_code == requests.codes.ok):
  jsondata = r.json()
  result = jsondata["result"]

#Result of RPC call
# result={
#    'count': 384982,
#    'status': 'OK'
# }
```

Example in Python

```
curl -X POST http://127.0.0.1:20206/json_rpc -d '{"jsonrpc":"2.0","id":"1","
    method":"getblockcount"}' -H 'Content-Type: application/json'
```

Example using curl

Chapter 2

Quick Overview

Method	Section	Description
getblockcount	3.2.1	Returns the currently synced height of the chain
get_info	3.2.2	Returns various info about the daemon and network
getblocktemplate	3.2.3	Return a block template (used for mining a block)
submitblock	3.2.4	Submits a mined block to the network
getlastblockheader	3.2.5	Returns the latest blockheader of the currently synced height
getblockheaderbyhash	3.2.6	Returns a blockheader from its hash
getblockheaderbytopoheight	3.2.7	Returns the blockheader from given topoheight
getblockheaderbyheight	3.2.8	Returns the blockheader from given height
getblock	3.2.9	Returns a block from its given hash
gettxpool	3.2.10	Returns a list of all the pending txhashes in the mempool
getheight	3.3.1	Returns the currently synced height and topoheight of the chain
gettransactions	3.3.2	Returns the specified transactions from either the blockchain or mempool
sendrawtransaction	3.3.3	Send a raw transaction to the network
is_key_image_spent	3.3.4	Checks if one of the supplied key images has been spent

Daemon RPC methods

Method	Section	Description
getaddress	4.2.1	Return wallet address
getbalance	4.2.2	Return wallet balance
getheight	4.2.3	Return wallet height
transfer	4.2.4	Send Dero to another wallet address
transfer_split	4.2.5	Same as transfer
get_bulk_payments	4.2.6	Return payments with requested paymentIDs and filtered by height requested
query_key	4.2.7	Return seed or view key
make_integrated_address	4.2.8	Generate integrated address with specified payment IDs
split_integrated_address	4.2.9	Split integrated address into standard wallet address and payment ID
get_transfer_by_txid	4.2.10	Get transfer information by ID
get_transfers	4.2.11	Get all out/ingoing transactions from a wallet

Wallet RPC methods

Chapter 3

DERO Daemon RPC Interface

3.1 Introduction

When launched, the Dero daemon automatically starts the RPC server interface at port 20206. You can change the port by using the rpc-bind parameter:

```
.\dero-wallet-cli.exe --rpc-bind=127.0.0.1:20206
```

3.2 Methods via POST

Most RPC methods work by issuing HTTP POST requests and sending the parameters in the payload.

```
#construct payload
payload = {'jsonrpc': '2.0', 'id': '1', 'method': 'getheight'}

#send request
r = requests.post('http://127.0.0.1:20206/json_rpc', json=payload, headers={'
    Connection': 'close'})

#read response
if (r.status_code == requests.codes.ok):
        jsondata = r.json()
        result = jsondata["result"]

#result = {
# 'height': 416543
#}
```

Example Python script

3.2.1 getblockcount

The method "getblockcount" returns the height of the (currently synced) chain. This is also the currenty unstable height. This method is called without parameters.

Result	Type	Description
"count"	uint64	The current blockheight
"status"	String	Always returns "OK"

Results of getblockcount

```
payload = {'jsonrpc': '2.0', 'id':'1', 'method': 'getblockcount'}

result={
  'count': 384982,
  'status': 'OK'
}
```

Example of getblockcount output

3.2.2 get_info

The method "get_info" returns various info about the daemon and the state of the network. This method has no parameters.

Result	Type	Description
"alt_blocks_count"	uint64	Unused
"difficulty"	uint64	The current difficulty
"grey_peerlist_size"	uint64	Unused
"height"	int64	Current blockchain height
"stableheight"	int64	Current stable height
"topoheight"	int64	Current topoheight
"averageblocktime50"	float32	Average blocktime of the last 50 blocks
"incoming_connections_count"	uint64	Unused
"outgoing_connections_count"	uint64	Unused
"target"	uint64	Target blocktime in seconds
"target_height"	uint64	Unused
"testnet"	bool	Indicates if this is a testnet
"top_block_hash"	string	Block ID of the newest block
"tx_count"	uint64	Unused
"tx_pool_size"	uint64	Number of pending transactions in the mempool
"dynamic_fee_per_kb"	uint64	Transaction fee
"total_supply"	uint64	Total coin supply (minus premine)
"median_block_Size"	uint64	Max blocksize in bytes (currently 1.25 MB)
"white_peerlist_size"	uint64	Unused
"version"	string	Daemon version
"status"	string	Always returns "OK"

Results of get_info

```
payload = {'jsonrpc': '2.0', 'id':'1', 'method': 'get_info'}

result = {
'alt_blocks_count': 0,
'difficulty': 2053657306,
'grey_peerlist_size': 0,
'height': 406525,
'stableheight': 406510,
'topoheight': 579299,
'averageblocktime50': 6,
'incoming_connections_count': 0,
'outgoing_connections_count': 0,
'target': 12,
'target_height': 0,
'testnet': False,
'top_block_hash': '4
   db40b72dba3b179dd10ecf7e90d55934509ee17c3b6de57bca5fefa3237315e',
'tx_count': 0,
'tx_pool_size': 0,
'dynamic_fee_per_kb': 1000000000,
'total_supply': 3393288,
'median_block_size': 1310720,
'white_peerlist_size': 0,
'version': '2.0.1-3.alpha.atlantis+04072018',
'status': 'OK'
}
```

Example of get_info output

3.2.3 getblocktemplate

Return a block template (used for mining a block).

Parameter	Type	Description
"wallet_address"	string	Hexdecimal string of the wallet address which receives the mining reward
"reserve_size"	uint64	should be > 0 and < 255

Parameters for getblocktemplate

Result	Type	Description
"blocktemplate_blob"	string	Hexdecimal string of the blocktemplate data
"blockhashing_blob"	string	Returns the getwork style blob, ready for either submitting the block or doing Pow Calculations
"expected_reward"	uint64	Always returns 0, not implemented yet
"difficulty"	uint64	The difficulty of the block
"height"	uint64	The height of the block
"prev_hash"	string	The hash of the previous block
"reserved_offset"	uint64	Returns the byte position of the extra nonce in the blockhashing blob
"epoch"	uint64	The expiry time of this block
"status"	string	Always returns "OK"

Results of getblocktemplate

```
 payload = {'jsonrpc': '2.0', 'id':'1', 'method': 'getblocktemplate', "params":{
     "wallet_address":"dERokSea2psYGJ...", "reserve_size":10}}

result = {
'blocktemplate_blob': '02028ed7b2db05cb99daea0000000000000000...',
'blockhashing_blob': '02028eab6c5b00fe0311b61b2fd324cfb7b01f...',
'expected_reward': 0,
'difficulty': 1804159513,
'height': 392836,
'prev_hash': 'a7e861a0a3362e9eb713f4cc1f3da04952914636107266ee5dbbfe...',
'reserved_offset': 43,
'epoch': 1533848474,
'status': 'OK'
}
```

Example of getblocktemplate output

3.2.4 submitblock

Submits a processed blocktemplate_blob and blockhashing_blob to the daemon. The parameter is unnamed as the data is transmitted as array and accessed by-position
(see https://www.jsonrpc.org/specification#parameter_structures).

Parameter	Type	Description
	[ ]string	Array (without name) containing the processed blocktemplate_blob and blockhashing_blob in hex

Parameters for submitblock

Result	Type	Description
"blid"	string	The new Block ID
"status"	string	Returns "OK" if ok, otherwise an error message e.g. "Could NOT decode block", "REJECTED" etc.

Results of submitblock

```
payload = {'jsonrpc': '2.0', 'id':'1', 'method': 'submitblock', "params":
    blockDatainHEX}

Answer: {
'blid': 591042,
'status': 'OK'
}
```

Example of submitblock output

3.2.5 getlastblockheader

The method "getlastblockheader" returns the latest blockheader of the (currently synced) chain. This is equal to the top unstable height. This method is called without parameters.

Result	Type	Description
"block_header"	block_header	The block header
"status"	string	Always returns "OK"

Results of getlastblockheader

Key	Type	Description
"depth"	int64	The difference between the current blockchain height and the height of this block
"difficulty"	string	Difficulty of this block
"hash"	string	Hash of the block, serves as block ID
"height"	int64	The height of the block
"topoheight"	int64	The topheight of the block
"major_version"	uint64	Current Atlanis blocks have major version "2"
"minor_version"	uint64	Current Atlanis blocks have minor version "2"
"nonce"	string	The block nonce
"orphan_status"	bool	Indicates if a block is orphaned, should not happen with Atlantis
"syncblock"	bool	Indicates wether a block is a syncblock, that is a single stable block used for rebuilding the chain
"txcount"	int64	Number of transactions included in the block
"reward"	uint64	The reward (base + fees) for this block
"tips"	[]string	Block IDs of the parent of this block
"timestamp"	uint64	The timestamp of the block

JSON structure block_header

```
payload = {'jsonrpc': '2.0', 'id':'1', 'method': 'getlastblockheader'}

result = {
'block_header':
{
  'depth': 0,
  'difficulty': '2049682764',
  'hash': '5d49e6e9eea00a7c066e14f4ebc05473b39306d8ba05d741f3955658f17322e2',
  'height': 397287,
  'topoheight': 563285,
  'major_version': 2,
  'minor_version': 2,
  'nonce': 806404222,
  'orphan_status': False,
  'syncblock': False,
  'txcount': 0,
  'reward': 1496206893845,
 'tips': ['7527285409f6b7133153a25a1f2f9848dc95758a0be7e9d79365ac3854841644', '
    e9f065645839501f43ce1aae95e491d19ae542fabd99a33b0497938627206477c'],
  'timestamp': 1533702138
},
'status': 'OK'
}
```

Example of getlastblockheader output

3.2.6 getblockheaderbyhash

The method "getblockheaderbyhash" returns the blockheader for the supplied blocks hash.

Parameter	Type	Description
"hash"	string	The hash (block ID)

Parameters for getblockheaderbyhash

Result	Type	Description
"block_header"	block_header	The block header, description of this structure can be found in 3.2.5
"status"	string	Always returns "OK"

Results of getblockheaderbyhash

```
payload = {'jsonrpc': '2.0', 'id': '1', 'method': 'getblockheaderbyhash', "
    params": {"hash": '7b6113e05eb72f7ed4231079f97f2708...'}}

result =
{'block_header':
{
  'depth': 112634,
  'difficulty': '1938163580',
  'hash': '7b6113e05eb72f7ed4231079f97f2708c6c0eff8cb8a0eb2c079b7ebcdd90157',
  'height': 320166,
  'topoheight': 432401,
  'major_version': 2,
  'minor_version': 2,
  'nonce': 2754379831,
  'orphan_status': False,
  'syncblock': True,
  'txcount': 0,
  'reward': 1515705893956,
  'tips': ['59a67a563e768da4e8d30940485c37fa22beea1da80c8850da6e690a2467ae5a'],
  'timestamp': 1532790081
},
'status': 'OK'
}
```

Example of getblockheaderbyhash output

3.2.7 getblockheaderbytopoheight

The method "getblockheaderbytopoheight" returns the blockheader for the supplied topoheight.

Parameter	Type	Description
"topheight"	uint64	The topoheight

Parameters for getblockheaderbytopoheight

Result	Type	Description
"block_header"	block_header	The block header, description of this structure can be found in 3.2.5
"status"	string	Always returns "OK"

Results of getblockheaderbytopoheight

```
payload = {'jsonrpc': '2.0', 'id':'1', 'method': 'getblockheaderbytopoheight', "
    params":{ "topoheight":320000}}

result = {
'block_header':
{
  'depth': 185434,
  'difficulty': '1967934625',
  'hash': 'd4fb3bbba2e6c0260330169f0b7f947c8c470b10a4e2f37d27d3047fd74d4bea',
  'height': 247366,
  'topoheight': 320000,
  'major_version': 2,
  'minor_version': 2,
  'nonce': 3346301122,
  'orphan_status': False,
  'syncblock': False,
  'txcount': 0,
  'reward': 1027132841775,
  'basereward': 1027132841775,
  'tips': ['2cc6b73cc75b81accb9867befb72fff1b4031a44792607f21a9d9ed049baab92', '
     eb2258c6459e05966d88bcc6b46191910a512ffa17f7f813c480fe0682456f38'],
  'timestamp': 1531920078
},
'status': 'OK'
}
```

Example of getblockheaderbytopoheight output

3.2.8 getblockheaderbyheight

The method "getblockheaderbyheight" returns the blockheader for the supplied blockchain topo-height, it does the same as the getblockheaderbytopoheight function.

Parameter	Type	Description
"height"	uint64	The blockchain height

Parameters for getblockheaderbyheight

Result	Type	Description
"block_header"	block_header	The block header, description of this structure can be found in 3.2.5
"status"	string	Always returns "OK"

Results of getblockheaderbyheight

```
payload = {'jsonrpc': '2.0', 'id':'1', 'method': 'getblockheaderbyheight', "
    params":{ "height":320000}}

result = {
'block_header':
{
  'depth': 185434,
  'difficulty': '1967934625',
  'hash': 'd4fb3bbba2e6c0260330169f0b7f947c8c470b10a4e2f37d27d3047fd74d4bea',
  'height': 247366,
  'topoheight': 320000,
  'major_version': 2,
  'minor_version': 2,
  'nonce': 3346301122,
  'orphan_status': False,
  'syncblock': False,
  'txcount': 0,
  'reward': 1027132841775,
  'basereward': 1027132841775,
  'tips': ['2cc6b73cc75b81accb9867befb72fff1b4031a44792607f21a9d9ed049baab92', '
     eb2258c6459e05966d88bcc6b46191910a512ffa17f7f813c480fe0682456f38'],
  'timestamp': 1531920078
},
'status': 'OK'
}
```

Example of getblockheaderbyheight output

3.2.9 getblock

The method "getblock" returns the data of a block from either the given height or hash.

Parameter	Type	Description
"hash"	string	Hash (block ID) of the block
"height"	uint64	height of the block

Parameters for getblock

Result	Type	Description
"blob"	string	Hexdecimal blob of the block data
"json"	string	String with the block data in JSON format
"block_header"	block_header	The block header, description of this structure can be found in 3.2.5
"status"	string	Always returns "OK"

Results of getblock

```
payload = {'jsonrpc': '2.0', 'id':'1', 'method': 'getblock', "params":{ "hash":"
    ded835887cea53195ca4e1d294d5158c..."}}

result = {
'blob': '0202dbc5aedb059080245b6b6a71c24e06d48b00...',
'json':
{
  "major_version":2,
  "minor_version":2,
  "timestamp":1533780699,
  "nonce":1529118864,
  "miner_tx":
  {
  "version":2,
  "unlock_time":404016,
  "Vin":[{"Height":403956}],
  "Vout":[
  {
  "Amount":0,
  "Target":{"Key":"94507a4d702b85f601ea51a97..."}}],
  "Extra":"AYyPD2XX6QEB19Xx9J0V3+UpXxZHEcQ2rGENQVHR2738",
  "RctSignature":
  {
  "Message":"00000000000000...",
  "MixRing":null,
  "ECdhInfo":null,
  "OutPk":null,
  "Txid":"0000000000000000...",
  "BulletSigs":null,
  "MlsagSigs":null
  }
  },
  "tips":["d7fc7c9d060e26bd9ca222..."],
  "tx_hashes":null
}',
'block_header': see getlastblockheader

}
```

Example of getblockheaderbyheight output

3.2.10 gettxpool

The method "gettxpool" returns the tx hashes that are currently in the mempool. This method has no parameters.

Result	Type	Description
"tx"	[ ]string	Tx hashes that are in the mempool
"status"	string	Always returns "OK"

Results of gettxpool

```
payload = {'jsonrpc': '2.0', 'id':'1', 'method': 'gettxpool'}

#(If txpool empty)
Result = {
'status': 'OK'
}

#else
Result = {
'txs': ['d1a63adf3385f44a20f451a9c38a3b3f6ce0071876b32a3ff7de12b92ccba6c2', '0
    c237ebf3daaab36acde2cfbbad31e5d5435a114d691abeaf2b407d3c03560ac', '
    e5b3e7f49b911734196835fe9a0718fab5f17432aa69c55a035cdee2a1765a7d'],
'status': 'OK'
}
```

Example of gettxpool output

3.3 Methods via GET

For compability reasons, some RPC methods work using HTTP GET requests. Thereby the method name is part of the URL and the parameters are transmitted in the payload.

```python
#construct payload for HTTP request
payload = {"key_images": ['7ab535bac7d0900cc4b71362e7a3037fbdff203df728a99001b3ceea925f4cde']}

#send request
r = requests.get('http://127.0.0.1:20206/is_key_image_spent', json=payload, headers={'Connection': 'close'})

#read response object
  if (r.status_code == requests.codes.ok):
    jsondata = r.json()
    result = jsondata["result"]

#result = {
# 'spent_status': [1],
# 'status': 'OK'
#}
```

Example Python script

3.3.1 getheight

The method "getheight" returns the different heights of the blockchain. It is called without parameters.

Result	Type	Description
"height"	uint64	Current blockchain height
"stableheight"	int64	Current stable height
"topoheight"	int64	Current topoheight
"status"	string	Always returns "OK"

Results of gettxpool

```
r = requests.get('http://127.0.0.1:20206/getheight')

result = {
'height': 413281,
'stableheight': 413271,
'topoheight': 591277,
'status': 'OK'
}
```

Example of gettxpool output

3.3.2 gettransactions

The method "gettransactions" returns the transaction data for a list of transaction IDs in hex and JSON format.

Parameter	Type	Description
"txs_hashes"	[]string	The tx hashes
"decode_as_json"	uint64	Unused

Parameters for gettransactions

Result	Type	Description
"txs_as_hex"	[ ]string	Tx data as hex
"txs_as_json"	[ ]string	unused
"txs"	[ ]Tx_Related_Info	JSON structures of Tx data
"status"	string	"OK" or "TX NOT FOUND"

Results of gettransactions

```
payload = {"txs_hashes":['
    db408453b56de5e9973ce4b0fd46edd6b0460437602f84e2bd7861c2a0ca4468', '0
    c237ebf3daaab36acde2cfbbad31e5d5435a114d691abeaf2b407d3c03560ac']}
r = requests.get('http://http://127.0.0.1:20206/gettransactions', json=payload,
    headers={'Connection': 'close'})


#(If tx is not found)
Result = {
"txs_as_hex":[""],
"txs_as_json":null,
"txs":null,
"status":"TX NOT FOUND"}
}

#else
Result = {
'txs_as_hex': ['020003020005e9b032a2e501f28b10b6cc06...'],
'txs_as_json': None,
'txs': [ {
  'as_hex': '',
  'as_json': '',
  'block_height': 596325,
  'reward': 0,
  'ignored': False,
  'in_pool': False,
  'output_indices': [1532913],
  'tx_hash': '',
  'valid_block': '53
     c9fac4bee2e01c9576cb30e6deb7261fc71492fae32e14a042f917e506a857',
    'invalid_block': None,
    'ring': [...]
  ]
}
```

Example of gettransactions output

3.3.3 sendrawtransaction

The method "sendrawtransaction" takes a transaction as hex data and submits it into the mem-pool if valid. Various checks are performed to ensure the transaction is valid, however the boolean result fields are not used, so you have to check the "status" value.

Parameter	Type	Description
"tx_as_hex"	string	The tx data as hex

Parameters for sendrawtransaction

Result	Type	Description
"double_spend"	bool	Unused
"fee_too_low"	bool	Unused
"invalid_input"	bool	Unused
"invalid_output"	bool	Unused
"low_mixin"	bool	Unused
"not_rct"	bool	Unused
"not_relayed"	bool	Unused
"overspend"	bool	Unused
"too_big"	bool	Unused
"status"	string	"OK" if tx send, otherwise error message

Results of sendrawtransaction

```
payload = {"tx_as_hex": '0200030200e973029741f64eef6ad239ff00d168cb9f481ed4aff
    ...'}

#This transaction failed because it was already sent, double spent rejection
Answer: {
  "status":"Transaction
      e3a35c1e829c28a714beed5e59af5da064ade8f581d27add9918d22fc615a10f rejected
      by daemon, check daemon msgs",
  "double_spend":false,
  "fee_too_low":false,
  "invalid_input":false,
  "invalid_output":false,
  "low_mixin":false,
  "not_rct":false,
  "not_relayed":false,
  "overspend":false,
  "too_big":false,
  "string":""
}
```

Example of sendrawtransaction output

3.3.4 is_key_image_spent

The method "is_key_image_spent" is used to check the status of a list of key_images.

Parameter	Type	Description
"key_images"	[ ]string	The key images as hex

Parameters for is_key_image_spent

Result	Type	Description
"spent_status"	[ ]int	0 if okay, 1 spent in block chain, 2 spent in pool
"status"	string	"OK"

Results of is_key_image_spent

```
payload = {"key_images": ['7ab535bac7d0900cc4b71362e7a303...']}

result = {
  'spent_status': [1],
  'status': 'OK'
}
```

Example of is_key_image_spent output

Chapter 4

DERO Wallet RPC Interface

4.1 Introduction

To the use the RPC interface, the wallet has to be started with the rpc-server function enabled:

```
.\dero-wallet-cli.exe --rpc-server
```

If your machine is accessible from the outside, you should setup a rpc login to access the wallet:

```
.\dero-wallet-cli.exe --rpc-server --rpc-login=<username:password>
```

The default port number for the wallet is 20209, but this can be also changed:

```
.\dero-wallet-cli.exe --rpc-server --rpc-login=<username:password>  --rpc-bind
   =<127.0.0.1:20209>
```

Note: the Dero daemon has to also be started to sync up the wallet transactions.

4.2 Methods via POST

All wallet RPC methods work by issuing HTTP POST requests and sending the parameters in the payload.

```
payload = {'jsonrpc': '2.0', 'id': '1', 'method': 'getheight'}
r = requests.post('http://127.0.0.1:20209/json_rpc', json=payload, headers={'
   Connection': 'close'})
if (r.status_code == requests.codes.ok):
        jsondata = r.json()
        result = jsondata["result"]

result = {
  'height': 416543
}
```

Example Python script

4.2.1 getaddress

The method "getaddress" method is used to the address from the wallet. This method has no parameters.

Result	Type	Description
"address"	string	The wallet address

Results of getaddress

```
payload = {'jsonrpc': '2.0', 'id':'1', 'method': 'getaddress'}

Result = {
  'address': 'dERokSea2psYGJNCC3KFpTNSZJm5ZEbfbb2hVq...'
}
```

Example of getaddress output

4.2.2 getbalance

The method "getbalance" method is used to get the current balance and unlocked from the wallet. This method has no parameters.

Result	Type	Description
"balance"	uint64	The wallet balance
"unlocked_balance"	uint64	The unlocked balance

Results of getbalance

```
payload = {'jsonrpc': '2.0', 'id':'1', 'method': 'getbalance'}

Result = {
  'balance': 0,
  'unlocked_balance': 0
}
```

Example of getbalance output

4.2.3 getheight

The method "getheight" method is used to get the currently synced blockchain height from the wallet. This method has no parameters.

Result	Type	Description
"height"	uint64	The wallet height

Results of getheight

```
payload = {'jsonrpc': '2.0', 'id':'1', 'method': 'getheight'}

Result = {
  'height': 416543
}
```

Example of getheight output

4.2.4 transfer

The method "transfer" method is used to create one or multiple transactions from a list of destinations. The optionally returned tx_key can used to prove that the amount was sent to the address using the Dero block explorer.

Parameter	Type	Description
"destinations"	[ ]Destination	The transfer addresses & amounts
"fee"	uint64	Unused
"mixin"	uint64	The mixin (anonymity setting)
"unlock_time"	uint64	The block height when the payment shall be unlocked
"payment_id"	string	The payment id
"get_tx_key"	bool	If the tx key shall be returned
"priority"	uint64	Unused
"do_not_relay"	bool	Unused
"get_tx_hex"	bool	If the tx shall be returned as hex blob

Parameters of transfer

Result	Type	Description
"amount"	uint64	The amount of Dero to be send
"address"	string	The address where Dero shall be send

JSON "destinations" structure

Result	Type	Description
"fee"	uint64	The fee spent
"tx_key"	string	The tx key
"tx_hash"	string	The tx ID
"tx_blob"	string	The tx data in hex

Results of transfer

```
payload = {'jsonrpc': '2.0', 'id': '1', 'method': 'transfer', "params":{ "
    destinations":[{"amount":amount, "address":address}], "Mixin":6, "unlock_time
    ":0, "payment_id":payment_id, "get_tx_key":True, "priority":1.0, "
    do_not_relay":False, "get_tx_hex":True }}




Result = {
  'fee': 4500000000,    # 0,0045 Dero
  'tx_key': 'b8cd3a1f6e8675606b9637ced1f1df40afb531d71b0ca2044b10938e9023500f',
  'tx_hash': 'c5238d0b0638fc8d2a0ca9252ed6c41df894c27dbcac9cb6e050bc89a99062a8',
  'tx_blob': '026402020006ffec1d97c71d...'
}
```

Example of transfer output

4.2.5 transfer_split

The method "transfer_split" is equal to the "transfer" method in 4.2.4 and kept for compatibility reasons.

```
payload = {'jsonrpc': '2.0', 'id': '1', 'method': 'transfer_split',
"params": {"destinations": [{"amount": amount, "address": address}, {"amount":
    amount, "address": address}], "Mixin": 6, "unlock_time": 0, "payment_id":
    payment_id, "get_tx_key": True, "priority": 1.0, "do_not_relay": False,"
    get_tx_hex": False}}


result ={
'fee_list': [4500000000],
'amount_list': None,
'tx_key_list': ['85
    f6db6a5bd034056494bae5b3ed2a43abed9535aa8733c56d4c937cb013720a'],
'tx_hash_list': ['4
    f81195b866682051b2ffec0cc2af2884c696ab5ae92ab649dc7c2a7f6589b27'],
'tx_blob_list': None
}
```

Example of transfer_split output

4.2.6 get_bulk_payments

The method "get_bulk_payments" is used to get a bulk of transactions from a list of payment IDs. Note: The method aborts if atleast one payment ID is not found/invalid.

Parameter	Type	Description
"payment_ids"	[ ]string	List of payment IDs to be checked. If no payment ID is provided, all entries with any payment ID are returned.
"min_block_height"	uint64	Blockheight where to start searching

Parameters of get_bulk_payments

Result	Type	Description
"payments"	[ ]Transfer_Details	The details for the transactions

Results of get_bulk_payments

Result	Type	Description
"tx_hash"	string	The hash of the transaction
"payment_id"	string	The payment ID
"block_height"	uint64	The blockheight of the transaction
"amount"	uint64	The amount received
"unlock_time"	uint64	The blockheight where it will be unlocked

JSON structure Transfer_Details

```
payload = {'jsonrpc': '2.0', 'id': '1', 'method': 'get_bulk_payments', "params":
    { "payment_ids":payment_ids, "min_block_height":400}}



result = {
  'tx_hash': 'b6d200a5da00dac27f0f5020fc92870b96f583bc25f8d0f7ff3096f5d64f7e4a',
  'payment_id': '533
     f1a4dc4f0a15095d384fafab07187222b13c9f18c02e6a289e36a5d6a5049',
  'block_height': 421258,
  'amount': 983800000000,
  'unlock_time': 0
}
```

Example of get_bulk_payments output

4.2.7 query_key

The method "query_key" is used to query either the secret key in mnemonic form or the view_key in hex form from the wallet.

Parameter	Type	Description
"key_type"	string	Type of key to query, can be "mnemonic" or "view_key"

Parameters of query_key

Result	Type	Description
"key"	string	The queried key

Results for query_key

```
payload = {'jsonrpc': '2.0', 'id': '1', 'method': 'query_key', "params": { "
    key_type":"view_key"}}

result = {
  'key':'b4a7478974345c5555efce9bf98b1a7bffc7e735e85c378d4ad04366b7e49809'
}
```

Example of query_key output

4.2.8 make_integrated_address

The method "make_integrated_address" is used to combine a payment_id and normal wallet address to create an integrated address.

Parameter	Type	Description
"payment_id"	string	16 or 64 hex encoded payment ID

Parameters of make_integrated_address

Result	Type	Description
"integrated_address"	string	The integrated address
"payment_id"	string	The payment ID used

make_integrated_address

```
payload = {'jsonrpc': '2.0', 'id': '1', 'method': 'make_integrated_address', "
    params": { "payment_id":"533
    f1a4dc4f0a15095d384fafab07187222b13c9f18c02e6a289e36a5d6a5049"}}

Answer:{
'integrated_address': 'dERimcuo7YfYGJNCC3KFpTNSZJm5ZEbfbb2hVqm4Nn...',
'payment_id': '533f1a4dc4f0a15095d384fafab07187222b13c9f18c02e6a289e36a5d6a5049'
}
```

Example of make_integrated_address output

4.2.9 split_integrated_address

The method "split_integrated_address" is used to get the payment_id and normal wallet address from an integrated address.

Parameter	Type	Description
"integrated_address"	string	The integrated address

Parameters of split_integrated_address

Result	Type	Description
"standard_address"	string	The normal wallet address
"payment_id"	string	The payment ID used

Results for split_integrated_address

```
payload = {'jsonrpc': '2.0', 'id': '1', 'method': 'split_integrated_address', "
    params": {"integrated_address": "dERimcuo7YfYGJNCC3KFpTNSZJm5ZEbfbb2hViv..."
    }}

result = {
'standard_address': 'dERokSea2psYGJNCC3KFpTNSZJm5ZEbfbb2hVqm4Nns72s...',
'payment_id': '533f1a4dc4f0a15095d384fafab07187222b13c9f18c02e6a289e36a5d6a5049'
    }
}
```

Example of split_integrated_address output

4.2.10 get_transfer_by_txid

The method "get_transfer_by_txid" is used to get transaction details for a specific transaction ID.

Parameter	Type	Description
"txid"	string	Transaction ID

Parameters of get_transfer_by_txid

Result	Type	Description
"payments"	Transfer_Details	The transaction details

Results of get_transfer_by_txid
(The "Transfer_Details" structure can be found in 4.2.6)

```
payload = {'jsonrpc': '2.0', 'id': '1', 'method': 'get_transfer_by_txid', "
    params": {"txid": "
    b6d200a5da00dac27f0f5020fc92870b96f583bc25f8d0f7ff3096f5d64f7e4a"}}

result = {
'payments': {
'tx_hash': 'b6d200a5da00dac27f0f5020fc92870b96f583bc25f8d0f7ff3096f5d64f7e4a',
'payment_id': '533f1a4dc4f0a15095d384fafab07187222b13c9f18c02e6a289e36a5d6a5049'
    ,
'block_height': 421258,
'amount': 983800000000,
'unlock_time': 0,
'type': 'in'}
}
```

Example of get_transfer_by_txid output

4.2.11 get_transfers

The method "get_transfers" is used to get all out/ingoing transactions from a wallet. You can use the min_height and max_height parameters to narrow down the scope.

Parameter	Type	Description
"in"	bool	Get the incoming transfers
"out"	bool	Get the outgoing transfers
"pending"	bool	Unused
"failed"	bool	Unused
"pool"	bool	Unused
"filter_by_height"	bool	Unused
"min_height"	unint64	Minimal blockheight
"max_height"	unint64	Maximum blockheight (if 0, up to current block)

Parameters for get_transfers

Result	Type	Description
"in"	[ ]Transfer_Details	If requested, the ingoing transactions
"out"	[ ]Transfer_Details	If requested, the outgoing transactions
"pending"	[ ]Transfer_Details	Unused
"failed"	[ ]Transfer_Details	Unused
"pool"	[ ]Transfer_Details	Unused

Results of get_transfers
(The "Transfer_Details" structure can be found in 4.2.6)

```
# get all outgoing transactions
payload = {'jsonrpc': '2.0', 'id': '1', 'method': 'get_transfers',
 "params": {"In": False, "Out":True, "Min_Height":0, "Max_Height":0}}

result = {
'out': [{'tx_hash': '
   e72a3437676aca4a0621c794e9c9c9efdbe0146b53070a51aac6aacf97ade2f6',
'block_height': 422416,
'amount': 2165799999700,
'unlock_time': 0,
'type': 'out'
}]
```

Example output of get_transfers