**Bitcoin-cli Commands**

**== Blockchain ==**

getbestblockhash

getblock "blockhash" ( verbosity )

getblockchaininfo

getblockcount

getblockhash height

getblockheader "hash" ( verbose )

getchaintips

getchaintxstats ( nblocks blockhash )

getdifficulty

getmempoolancestors txid (verbose)

getmempooldescendants txid (verbose)

getmempoolentry txid

getmempoolinfo

getrawmempool ( verbose )

gettxout "txid" n ( include\_mempool )

gettxoutproof ["txid",...] ( blockhash )

gettxoutsetinfo

preciousblock "blockhash"

pruneblockchain

verifychain ( checklevel nblocks )

verifytxoutproof "proof"

**== Control ==**

getinfo

getmemoryinfo ("mode")

help ( "command" )

stop

uptime

**== Generating ==**

generate nblocks ( maxtries )

generatetoaddress nblocks address (maxtries)

**== Mining ==**

getblocktemplate ( TemplateRequest )

getmininginfo

getnetworkhashps ( nblocks height )

prioritisetransaction <txid> <dummy value> <fee delta>

submitblock "hexdata" ( "dummy" )

**== Network ==**

addnode "node" "add|remove|onetry"

clearbanned

disconnectnode "[address]" [nodeid]

getaddednodeinfo ( "node" )

getconnectioncount

getnettotals

getnetworkinfo

getpeerinfo

listbanned

ping

setban "subnet" "add|remove" (bantime) (absolute)

setnetworkactive true|false

**== Rawtransactions ==**

combinerawtransaction ["hexstring",...]

createrawtransaction [{"txid":"id","vout":n},...] {"address":amount,"data":"hex",...} ( locktime ) ( replaceable )

decoderawtransaction "hexstring"

decodescript "hexstring"

fundrawtransaction "hexstring" ( options )

getrawtransaction "txid" ( verbose )

sendrawtransaction "hexstring" ( allowhighfees )

signrawtransaction "hexstring" ( [{"txid":"id","vout":n,"scriptPubKey":"hex","redeemScript":"hex"},...] ["privatekey1",...] sighashtype )

**== Util ==**

createmultisig nrequired ["key",...]

estimatefee nblocks

estimatesmartfee conf\_target ("estimate\_mode")

signmessagewithprivkey "privkey" "message"

validateaddress "address"

verifymessage "address" "signature" "message"

**== Wallet ==**

abandontransaction "txid"

abortrescan

addmultisigaddress nrequired ["key",...] ( "account" )

addwitnessaddress "address"

backupwallet "destination"

bumpfee "txid" ( options )

dumpprivkey "address"

dumpwallet "filename"

getaccount "address"

getaccountaddress "account"

getaddressesbyaccount "account"

getbalance ( "account" minconf include\_watchonly )

getnewaddress ( "account" )

getrawchangeaddress

getreceivedbyaccount "account" ( minconf )

getreceivedbyaddress "address" ( minconf )

gettransaction "txid" ( include\_watchonly )

getunconfirmedbalance

getwalletinfo

importaddress "address" ( "label" rescan p2sh )

importmulti "requests" ( "options" )

importprivkey "privkey" ( "label" ) ( rescan )

importprunedfunds

importpubkey "pubkey" ( "label" rescan )

importwallet "filename"

keypoolrefill ( newsize )

listaccounts ( minconf include\_watchonly)

listaddressgroupings

listlockunspent

listreceivedbyaccount ( minconf include\_empty include\_watchonly)

listreceivedbyaddress ( minconf include\_empty include\_watchonly)

listsinceblock ( "blockhash" target\_confirmations include\_watchonly include\_removed )

listtransactions ( "account" count skip include\_watchonly)

listunspent ( minconf maxconf ["addresses",...] [include\_unsafe] [query\_options])

listwallets

lockunspent unlock ([{"txid":"txid","vout":n},...])

move "fromaccount" "toaccount" amount ( minconf "comment" )

removeprunedfunds "txid"

sendfrom "fromaccount" "toaddress" amount ( minconf "comment" "comment\_to" )

sendmany "fromaccount" {"address":amount,...} ( minconf "comment" ["address",...] replaceable conf\_target "estimate\_mode")

sendtoaddress "address" amount ( "comment" "comment\_to" subtractfeefromamount replaceable conf\_target "estimate\_mode")

setaccount "address" "account"

settxfee amount

signmessage "address" "message"

walletlock

walletpassphrase "passphrase" timeout

walletpassphrasechange "oldpassphrase" "newpassphrase"

Bitcoin-cli examples

== Blockchain ==

**getbestblockhash**

**ubuntu§philminer1**:**ç/.bitcoin**$ bitcoin-cli getbestblockhash

0000000000000000005f1b845229de3be6fcc3df24bcf1ed3677c4c8c5b699ff

**getblock "blockhash" ( verbosity )**

**ubuntu§philminer1**:**ç/.bitcoin**$ bitcoin-cli getblock 0000000000000000005f1b845229de3be6fcc3df24bcf1ed3677c4c8c5b699ff

`

"hash": "0000000000000000005f1b845229de3be6fcc3df24bcf1ed3677c4c8c5b699ff",

"confirmations": 1,

"strippedsize": 939232,

"size": 1174757,

"weight": 3992453,

"height": 498594,

"version": 536870912,

"versionHex": "20000000",

"merkleroot": "59bcc25ed7a6ceb92a6e2bc75eaecce1c245f3c9f0b39afb63b7e60ff8a6ef25",

"tx": ¡

"cc6cec756fcf16d5725af76b8b9c145259d471af09ad895e12d61fad17cba32f",

"1480480dbf27fb044697c1c52702de7496ae8cd2baeb06663213db2fb962f2cf", …

¿,

"time": 1512924189,

"mediantime": 1512922070,

"nonce": 801344741,

"bits": "1800b0ed",

"difficulty": 1590896927258.079,

"chainwork": "000000000000000000000000000000000000000000c5aa5602c1de72bdd6e48a",

"previousblockhash": "0000000000000000002025a8a800c729594bb9a1c55c54aadc6b324bd3d20136"

getblockchaininfo

**ubuntu§philminer1**:**ç/.bitcoin**$ bitcoin-cli getblockchaininfo

`

"chain": "main",

"blocks": 498595,

"headers": 498595,

"bestblockhash": "00000000000000000082feccc4aacd6b3ecea7fdd0f44872a1fbe5ffb4bfb036",

"difficulty": 1590896927258.079,

"mediantime": 1512922863,

"verificationprogress": 0.9999994914530475,

"chainwork": "000000000000000000000000000000000000000000c5abc86d0cead7de63c9b8",

"pruned": false,

"softforks": ¡

`

"id": "bip34",

"version": 2,

"reject": `

"status": true

ñ

ñ,

`

"id": "bip66",

"version": 3,

"reject": `

"status": true

ñ

ñ,

`

"id": "bip65",

"version": 4,

"reject": `

"status": true

ñ

ñ

¿,

"bip9\_softforks": `

"csv": `

"status": "active",

"startTime": 1462060800,

"timeout": 1493596800,

"since": 419328

ñ,

"segwit": `

"status": "active",

"startTime": 1479168000,

"timeout": 1510704000,

"since": 481824

ñ

ñ

ñ

**getblockcount**

**ubuntu§philminer1**:**ç/.bitcoin**$ bitcoin-cli getblockcount

498595

>>>10 minutes later

**ubuntu§philminer1**:**ç/.bitcoin**$ bitcoin-cli getblockcount

498598

**getblockhash height**

**ubuntu§philminer1**:**ç/.bitcoin**$ bitcoin-cli getblockhash

error code: -1

error message:

getblockhash height

Returns hash of block in best-block-chain at height provided.

Arguments:

1. height (numeric, required) The height index

Result:

"hash" (string) The block hash

Examples:

> bitcoin-cli getblockhash 1000

> curl --user myusername --data-binary '`"jsonrpc": "1.0", "id":"curltest", "method": "getblockhash", "params": ¡1000¿ ñ' -H 'content-type: text/plain;' http://127.0.0.1:8332/

**ubuntu§philminer1**:**ç/.bitcoin**$ bitcoin-cli getblockhash 1

00000000839a8e6886ab5951d76f411475428afc90947ee320161bbf18eb6048

**ubuntu§philminer1**:**ç/.bitcoin**$ bitcoin-cli getblockhash 2

000000006a625f06636b8bb6ac7b960a8d03705d1ace08b1a19da3fdcc99ddbd

**ubuntu§philminer1**:**ç/.bitcoin**$ bitcoin-cli getblockhash 3

0000000082b5015589a3fdf2d4baff403e6f0be035a5d9742c1cae6295464449

**ubuntu§philminer1**:**ç/.bitcoin**$ bitcoin-cli getblockhash 4

000000004ebadb55ee9096c9a2f8880e09da59c0d68b1c228da88e48844a1485

**ubuntu§philminer1**:**ç/.bitcoin**$ bitcoin-cli getblockhash 5

000000009b7262315dbf071787ad3656097b892abffd1f95a1a022f896f533fc

**ubuntu§philminer1**:**ç/.bitcoin**$

**getblockheader** "hash" ( verbose )

**ubuntu§philminer1**:**ç/.bitcoin**$ bitcoin-cli getblockheader 000000009b7262315dbf071787ad3656097b892abffd1f95a1a022f896f533fc

`

"hash": "000000009b7262315dbf071787ad3656097b892abffd1f95a1a022f896f533fc",

"confirmations": 498595,

"height": 5,

"version": 1,

"versionHex": "00000001",

"merkleroot": "63522845d294ee9b0188ae5cac91bf389a0c3723f084ca1025e7d9cdfe481ce1",

"time": 1231471428,

"mediantime": 1231470173,

"nonce": 2011431709,

"bits": "1d00ffff",

"difficulty": 1,

"chainwork": "0000000000000000000000000000000000000000000000000000000600060006",

"previousblockhash": "000000004ebadb55ee9096c9a2f8880e09da59c0d68b1c228da88e48844a1485",

"nextblockhash": "000000003031a0e73735690c5a1ff2a4be82553b2a12b776fbd3a215dc8f778d"

**getchaintips**

**ubuntu§philminer1**:**ç/.bitcoin**$ bitcoin-cli getchaintips

¡

`

"height": 498599,

"hash": "0000000000000000006dfdb5f40f55bc7689dcda49fad1b3a24a8d2ca319a844",

"branchlen": 0,

"status": "active"

ñ,

`

"height": 498269,

"hash": "00000000000000000061fd082c44ac1261572655e711b1efca6f586e47e35be4",

"branchlen": 1,

"status": "headers-only"

ñ,

`

"height": 497871,

"hash": "000000000000000000c314b3c7de6a7ee8251ebbb55e9a34181afda0b3bf799b",

"branchlen": 1,

"status": "valid-fork"

ñ,

`

"height": 497373,

"hash": "0000000000000000000d450f4d1ccbc5107f1eaa98284c2c87b6a0702c49c439",

"branchlen": 1,

"status": "valid-fork"

ñ

¿

**ubuntu§philminer1**:**ç/.bitcoin**$

**getchaintxstats** ( nblocks blockhash )

**ubuntu§philminer1**:**ç/.bitcoin**$ bitcoin-cli getchaintxstats

`

"time": 1512925382,

"txcount": 280414353,

"txrate": 4.064761657629542

getdifficulty

**ubuntu§philminer1**:**ç/.bitcoin**$ bitcoin-cli getdifficulty

1590896927258.079

getmempoolancestors txid (verbose)

getmempooldescendants txid (verbose)

getmempoolentry txid

**getmempoolinfo**

**ubuntu§philminer1**:**ç/.bitcoin**$ bitcoin-cli getmempoolinfo

`

"size": 74800,

"bytes": 80992306,

"usage": 206710992,

"maxmempool": 300000000,

"mempoolminfee": 0.00002593

ñ

getrawmempool ( verbose )

¡

"51c2612d5135ec3df1fdc89392272614009610bb90fb828aab0cdc67b4a095ec",

"c649e91282d067f50a00b2fc118e78acfc003f03388e5249b81132728bb38206",

"8a0394e2c8dfbf5ecbbc632cc0216a638727cbf44645586fa2fe27692c339437",

…..

**gettxout "txid"** n ( include\_mempool )

**ubuntu§philminer1**:**ç/.bitcoin**$ bitcoin-cli gettxout

error code: -1

error message:

gettxout "txid" n ( include\_mempool )

Returns details about an unspent transaction output.

Arguments:

1. "txid" (string, required) The transaction id

2. "n" (numeric, required) vout number

3. "include\_mempool" (boolean, optional) Whether to include the mempool. Default: true. Note that an unspent output that is spent in the mempool won't appear.

Result:

`

"bestblock" : "hash", (string) the block hash

"confirmations" : n, (numeric) The number of confirmations

"value" : x.xxx, (numeric) The transaction value in BTC

"scriptPubKey" : ` (json object)

"asm" : "code", (string)

"hex" : "hex", (string)

"reqSigs" : n, (numeric) Number of required signatures

"type" : "pubkeyhash", (string) The type, eg pubkeyhash

"addresses" : ¡ (array of string) array of bitcoin addresses

"address" (string) bitcoin address

,...

¿

ñ,

"coinbase" : true°false (boolean) Coinbase or not

ñ

Examples:

Get unspent transactions

> bitcoin-cli listunspent

View the details

> bitcoin-cli gettxout "txid" 1

As a json rpc call

> curl --user myusername --data-binary '`"jsonrpc": "1.0", "id":"curltest", "method": "gettxout", "params": ¡"txid", 1¿ ñ' -H 'content-type: text/plain;' http://127.0.0.1:8332/

**listunspent**

**ubuntu§philminer1**:**ç/.bitcoin**$ bitcoin-cli listunspent

¡

**gettxoutproof** ["txid",...] ( blockhash )

**ubuntu§philminer1**:**ç/.bitcoin**$ bitcoin-cli gettxoutproof

error code: -1

error message:

gettxoutproof ¡"txid",...¿ ( blockhash )

Returns a hex-encoded proof that "txid" was included in a block.

NOTE: By default this function only works sometimes. This is when there is an

unspent output in the utxo for this transaction. To make it always work,

you need to maintain a transaction index, using the -txindex command line option or

specify the block in which the transaction is included manually (by blockhash).

Arguments:

1. "txids" (string) A json array of txids to filter

¡

"txid" (string) A transaction hash

,...

¿

2. "blockhash" (string, optional) If specified, looks for txid in the block with this hash

Result:

"data" (string) A string that is a serialized, hex-encoded data for the proof.

**Gettxoutsetinfo**

(Note: this command took a minute to return a result)

**ubuntu§philminer1**:**ç/.bitcoin**$ bitcoin-cli gettxoutsetinfo

`

"height": 498601,

"bestblock": "0000000000000000008f35b6b8fadbaf24aaba042e07bed3c277331d4e042318",

"transactions": 26776596,

"txouts": 58390261,

"bogosize": 4390301158,

"hash\_serialized\_2": "45b1cb293531227b42d10f4a39ef0b9b73bf39fe9589a84e580caf5f8361846c",

"disk\_size": 3302233293,

"total\_amount": 16732361.08620595

ñ

**preciousblock "blockhash"**

**ubuntu§philminer1**:**ç/.bitcoin**$ bitcoin-cli preciousblock

preciousblock "blockhash"

Treats a block as if it were received before others with the same work.

A later preciousblock call can override the effect of an earlier one.

The effects of preciousblock are not retained across restarts.

Arguments:

1. "blockhash" (string, required) the hash of the block to mark as precious

Result:

Examples:

> bitcoin-cli preciousblock "blockhash"

> curl --user myusername --data-binary '`"jsonrpc": "1.0", "id":"curltest", "method": "preciousblock", "params": ¡"blockhash"¿ ñ' -H 'content-type: text/plain;' http://127.0.0.1:8332/

**pruneblockchain**

**ubuntu§philminer1**:**ç/.bitcoin**$ bitcoin-cli help pruneblockchain

pruneblockchain

Arguments:

1. "height" (numeric, required) The block height to prune up to. May be set to a discrete height, or a unix timestamp

to prune blocks whose block time is at least 2 hours older than the provided timestamp.

Result:

n (numeric) Height of the last block pruned.

Examples:

> bitcoin-cli pruneblockchain 1000

> curl --user myusername --data-binary '`"jsonrpc": "1.0", "id":"curltest", "method": "pruneblockchain", "params": ¡1000¿ ñ' -H 'content-type: text/plain;' http://127.0.0.1:8332/

verifychain ( checklevel nblocks )

**ubuntu§philminer1**:**ç/.bitcoin**$ bitcoin-cli verifychain

true

**ubuntu§philminer1**:**ç/.bitcoin**$ bitcoin-cli verifychain 3

true

**ubuntu§philminer1**:**ç/.bitcoin**$ bitcoin-cli verifychain 5

true

**verifytxoutproof "proof"**

verifytxoutproof "proof"

Verifies that a proof points to a transaction in a block, returning the transaction it commits to

and throwing an RPC error if the block is not in our best chain

Arguments:

1. "proof" (string, required) The hex-encoded proof generated by gettxoutproof

Result:

¡"txid"¿ (array, strings) The txid(s) which the proof commits to, or empty array if the proof is invalid

**== Control ==**

**getinfo**

**ubuntu§philminer1**:**ç/.bitcoin**$ bitcoin-cli getinfo

`

"deprecation-warning": "WARNING: getinfo is deprecated and will be fully removed in 0.16. Projects should transition to using getblockchaininfo, getnetworkinfo, and getwalletinfo before upgrading to 0.16",

"version": 150100,

"protocolversion": 70015,

"walletversion": 139900,

"balance": 0.00000000,

"blocks": 498603,

"timeoffset": -1,

"connections": 98,

"proxy": "",

"difficulty": 1590896927258.079,

"testnet": false,

"keypoololdest": 1511664417,

"keypoolsize": 2000,

"unlocked\_until": 0,

"paytxfee": 0.00000000,

"relayfee": 0.00001000,

"errors": ""

ñ

**getblockchaininfo (listed above under blockchain)**

**getnetworkinfo**

**ubuntu§philminer1**:**ç/.bitcoin**$ bitcoin-cli getnetworkinfo

`

"version": 150100,

"subversion": "/Satoshi:0.15.1/",

"protocolversion": 70015,

"localservices": "000000000000000d",

"localrelay": true,

"timeoffset": -1,

"networkactive": true,

"connections": 106,

"networks": ¡

`

"name": "ipv4",

"limited": false,

"reachable": true,

"proxy": "",

"proxy\_randomize\_credentials": false

ñ,

`

"name": "ipv6",

"limited": false,

"reachable": true,

"proxy": "",

"proxy\_randomize\_credentials": false

ñ,

`

"name": "onion",

"limited": true,

"reachable": false,

"proxy": "",

"proxy\_randomize\_credentials": false

ñ

¿,

"relayfee": 0.00001000,

"incrementalfee": 0.00001000,

"localaddresses": ¡

`

"address": "154.126.4245",

"port": 8333,

"score": 44464

ñ

¿,

"warnings": ""

ñ

getwalletinfo

**ubuntu§philminer1**:**ç/.bitcoin**$ bitcoin-cli getwalletinfo

`

"walletname": "wallet.dat",

"walletversion": 19800,

"balance": 0.00000000,

"unconfirmed\_balance": 0.00000000,

"immature\_balance": 0.00000000,

"txcount": 0,

"keypoololdest": 141162347,

"keypoolsize": 1000,

"keypoolsize\_hd\_internal": 1000,

"unlocked\_until": 0,

"paytxfee": 0.00000000,

"hdmasterkeyid": "########"

ñ

getmemoryinfo ("mode")

1. "mode" determines what kind of information is returned. This argument is optional, the default mode is "stats".

- "stats" returns general statistics about memory usage in the daemon.

- "mallocinfo" returns an XML string describing low-level heap state (only available if compiled with glibc 2.10+).

Result (mode "stats"):

`

"locked": ` (json object) Information about locked memory manager

"used": xxxxx, (numeric) Number of bytes used

"free": xxxxx, (numeric) Number of bytes available in current arenas

"total": xxxxxxx, (numeric) Total number of bytes managed

"locked": xxxxxx, (numeric) Amount of bytes that succeeded locking. If this number is smaller than total, locking pages failed at some point and key data could be swapped to disk.

"chunks\_used": xxxxx, (numeric) Number allocated chunks

"chunks\_free": xxxxx, (numeric) Number unused chunks

ñ

ñ

**ubuntu§philminer1**:**ç/.bitcoin**$ bitcoin-cli getmemoryinfo

`

"locked": `

"used": 0,

"free": 65536,

"total": 65536,

"locked": 65536,

"chunks\_used": 0,

"chunks\_free": 1

ñ

ñ

**help** ( "command" ) – list all bitcoin-cli commands

**stop** - stop the bitcoin server

**uptime**

Returns the total uptime of the server.

Result:

ttt (numeric) The number of seconds that the server has been running

Examples:

> bitcoin-cli uptime

> curl --user myusername --data-binary '`"jsonrpc": "1.0", "id":"curltest", "method": "uptime", "params": ¡¿ ñ' -H 'content-type: text/plain;' http://127.0.0.1:8332/

**ubuntu§philminer1**:**ç/.bitcoin**$ bitcoin-cli uptime

1263047

**ubuntu§philminer1**:**ç/.bitcoin**$

**== Generating ==**

**generate nblocks** ( maxtries )

generate nblocks ( maxtries )

Mine up to nblocks blocks immediately (before the RPC call returns) to an address in the wallet.

Arguments:

1. nblocks (numeric, required) How many blocks are generated immediately.

2. maxtries (numeric, optional) How many iterations to try (default = 1000000).

Result:

¡ blockhashes ¿ (array) hashes of blocks generated

Examples:

Generate 11 blocks

> bitcoin-cli generate 11

**generatetoaddress** nblocks address (maxtries)

generatetoaddress nblocks address (maxtries)

Mine blocks immediately to a specified address (before the RPC call returns)

Arguments:

1. nblocks (numeric, required) How many blocks are generated immediately.

2. address (string, required) The address to send the newly generated bitcoin to.

3. maxtries (numeric, optional) How many iterations to try (default = 1000000).

Result:

¡ blockhashes ¿ (array) hashes of blocks generated

Examples:

Generate 11 blocks to myaddress

> bitcoin-cli generatetoaddress 11 "myaddress"

**ubuntu§philminer1**:**ç/.bitcoin**$

**== Mining ==**

**getblocktemplate** ( TemplateRequest )

getblocktemplate ( TemplateRequest )

If the request parameters include a 'mode' key, that is used to explicitly select between the default 'template' request or a 'proposal'.

It returns data needed to construct a block to work on.

For full specification, see BIPs 22, 23, 9, and 145:

https://github.com/bitcoin/bips/blob/master/bip-0022.mediawiki

https://github.com/bitcoin/bips/blob/master/bip-0023.mediawiki

https://github.com/bitcoin/bips/blob/master/bip-0009.mediawiki£getblocktemplate\_changes

https://github.com/bitcoin/bips/blob/master/bip-0145.mediawiki

Arguments:

1. template\_request (json object, optional) A json object in the following spec

`

"mode":"template" (string, optional) This must be set to "template", "proposal" (see BIP 23), or omitted

"capabilities":¡ (array, optional) A list of strings

"support" (string) client side supported feature, 'longpoll', 'coinbasetxn', 'coinbasevalue', 'proposal', 'serverlist', 'workid'

,...

¿,

"rules":¡ (array, optional) A list of strings

"support" (string) client side supported softfork deployment

,...

¿

ñ

Result:

`

"version" : n, (numeric) The preferred block version

"rules" : ¡ "rulename", ... ¿, (array of strings) specific block rules that are to be enforced

"vbavailable" : ` (json object) set of pending, supported versionbit (BIP 9) softfork deployments

"rulename" : bitnumber (numeric) identifies the bit number as indicating acceptance and readiness for the named softfork rule

,...

ñ,

"vbrequired" : n, (numeric) bit mask of versionbits the server requires set in submissions

"previousblockhash" : "xxxx", (string) The hash of current highest block

"transactions" : ¡ (array) contents of non-coinbase transactions that should be included in the next block

`

"data" : "xxxx", (string) transaction data encoded in hexadecimal (byte-for-byte)

"txid" : "xxxx", (string) transaction id encoded in little-endian hexadecimal

"hash" : "xxxx", (string) hash encoded in little-endian hexadecimal (including witness data)

"depends" : ¡ (array) array of numbers

n (numeric) transactions before this one (by 1-based index in 'transactions' list) that must be present in the final block if this one is

,...

¿,

"fee": n, (numeric) difference in value between transaction inputs and outputs (in Satoshis); for coinbase transactions, this is a negative Number of the total collected block fees (ie, not including the block subsidy); if key is not present, fee is unknown and clients MUST NOT assume there isn't one

"sigops" : n, (numeric) total SigOps cost, as counted for purposes of block limits; if key is not present, sigop cost is unknown and clients MUST NOT assume it is zero

"weight" : n, (numeric) total transaction weight, as counted for purposes of block limits

"required" : true°false (boolean) if provided and true, this transaction must be in the final block

ñ

,...

¿,

"coinbaseaux" : ` (json object) data that should be included in the coinbase's scriptSig content

"flags" : "xx" (string) key name is to be ignored, and value included in scriptSig

ñ,

"coinbasevalue" : n, (numeric) maximum allowable input to coinbase transaction, including the generation award and transaction fees (in Satoshis)

"coinbasetxn" : ` ... ñ, (json object) information for coinbase transaction

"target" : "xxxx", (string) The hash target

"mintime" : xxx, (numeric) The minimum timestamp appropriate for next block time in seconds since epoch (Jan 1 1970 GMT)

"mutable" : ¡ (array of string) list of ways the block template may be changed

"value" (string) A way the block template may be changed, e.g. 'time', 'transactions', 'prevblock'

,...

¿,

"noncerange" : "00000000ffffffff",(string) A range of valid nonces

"sigoplimit" : n, (numeric) limit of sigops in blocks

"sizelimit" : n, (numeric) limit of block size

"weightlimit" : n, (numeric) limit of block weight

"curtime" : ttt, (numeric) current timestamp in seconds since epoch (Jan 1 1970 GMT)

"bits" : "xxxxxxxx", (string) compressed target of next block

"height" : n (numeric) The height of the next block

ñ

Examples:

> bitcoin-cli getblocktemplate

> curl --user myusername --data-binary '`"jsonrpc": "1.0", "id":"curltest", "method": "getblocktemplate", "params": ¡¿ ñ' -H 'content-type: text/plain;' http://127.0.0.1:8332/

**ubuntu§philminer1**:**ç/.bitcoin**$ bitcoin-cli getblocktemplate

`

"capabilities": ¡

"proposal"

¿,

"version": 536870912,

"rules": ¡

"csv",

"segwit"

¿,

"vbavailable": `

ñ,

"vbrequired": 0,

"previousblockhash": "000000000000000000ace3e3dfcd0a4b39d14b68decb86bff70842d9f139e8f7",

"transactions": ¡

"data": "",

"txid": "b97e2ff0ee288e9b33784bc2b3d2d10e618c9415bd8c6c6490479f4c458efe76",

"hash": "b97e2ff0ee288e9b33784bc2b3d2d10e618c9415bd8c6c6490479f4c458efe76",

"depends": ¡

¿,

"fee": 122400,

"sigops": 8,

"weight": 1024

…

**getmininginfo**

**ubuntu§philminer1**:**ç/.bitcoin**$ bitcoin-cli getmininginfo

`

"blocks": 498608,

"currentblockweight": 3995300,

"currentblocktx": 719,

"difficulty": 1590896927258.079,

"errors": "",

"networkhashps": 1.302859369792761e+19,

"pooledtx": 76211,

"chain": "main"

ñ

**getnetworkhashps** ( nblocks height )

Returns the estimated network hashes per second based on the last n blocks.

Pass in ¡blocks¿ to override £ of blocks, -1 specifies since last difficulty change.

Pass in ¡height¿ to estimate the network speed at the time when a certain block was found.

Arguments:

1. nblocks (numeric, optional, default=120) The number of blocks, or -1 for blocks since last difficulty change.

2. height (numeric, optional, default=-1) To estimate at the time of the given height.

Result:

x (numeric) Hashes per second estimated

Examples:

> bitcoin-cli getnetworkhashps

> curl --user myusername --data-binary '`"jsonrpc": "1.0", "id":"curltest", "method": "getnetworkhashps", "params": ¡¿ ñ' -H 'content-type: text/plain;' http://127.0.0.1:8332/

**ubuntu§philminer1**:**ç/.bitcoin**$ bitcoin-cli getnetworkhashps

1.302859369792761e+19

**ubuntu§philminer1**:**ç/.bitcoin**$

**prioritisetransaction** <txid> <dummy value> <fee delta>

prioritisetransaction <txid> <dummy value> <fee delta>

Accepts the transaction into mined blocks at a higher (or lower) priority

Arguments:

1. "txid" (string, required) The transaction id.

2. dummy (numeric, optional) API-Compatibility for previous API. Must be zero or null.

DEPRECATED. For forward compatibility use named arguments and omit this parameter.

3. fee\_delta (numeric, required) The fee value (in satoshis) to add (or subtract, if negative).

The fee is not actually paid, only the algorithm for selecting transactions into a block

considers the transaction as it would have paid a higher (or lower) fee.

Result:

true (boolean) Returns true

Examples:

> bitcoin-cli prioritisetransaction "txid" 0.0 10000

> curl --user myusername --data-binary '`"jsonrpc": "1.0", "id":"curltest", "method": "prioritisetransaction", "params": ¡"txid", 0.0, 10000¿ ñ' -H 'content-type: text/plain;' http://127.0.0.1:8332/

**submitblock** "hexdata" ( "dummy" )

**ubuntu§philminer1**:**ç/.bitcoin**$ bitcoin-cli help submitblock

submitblock "hexdata" ( "dummy" )

Attempts to submit new block to network.

See https://en.bitcoin.it/wiki/BIP\_0022 for full specification.

Arguments

1. "hexdata" (string, required) the hex-encoded block data to submit

2. "dummy" (optional) dummy value, for compatibility with BIP22. This value is ignored.

Result:

Examples:

> bitcoin-cli submitblock "mydata"

> curl --user myusername --data-binary '`"jsonrpc": "1.0", "id":"curltest", "method": "submitblock", "params": ¡"mydata"¿ ñ' -H 'content-type: text/plain;' http://127.0.0.1:8332/

**== Network ==**

**addnode** "node" "add|remove|onetry"

Attempts to add or remove a node from the addnode list.

Or try a connection to a node once.

Arguments:

1. "node" (string, required) The node (see getpeerinfo for nodes)

2. "command" (string, required) 'add' to add a node to the list, 'remove' to remove a node from the list, 'onetry' to try a connection to the node once

Examples:

> bitcoin-cli addnode "192.168.0.6:8333" "onetry"

> curl --user myusername --data-binary '`"jsonrpc": "1.0", "id":"curltest", "method": "addnode", "params": ¡"192.168.0.6:8333", "onetry"¿ ñ' -H 'content-type: text/plain;' http://127.0.0.1:8332/

**clearbanned** --- clear all banned IP addresses

**disconnectnode** "[address]" [nodeid]

Immediately disconnects from the specified peer node.

Strictly one out of 'address' and 'nodeid' can be provided to identify the node.

To disconnect by nodeid, either set 'address' to the empty string, or call using the named 'nodeid' argument only.

Arguments:

1. "address" (string, optional) The IP address/port of the node

2. "nodeid" (number, optional) The node ID (see getpeerinfo for node IDs)

Examples:

> bitcoin-cli disconnectnode "192.168.0.6:8333"

> bitcoin-cli disconnectnode "" 1

> curl --user myusername --data-binary '`"jsonrpc": "1.0", "id":"curltest", "method": "disconnectnode", "params": ¡"192.168.0.6:8333"¿ ñ' -H 'content-type: text/plain;' http://127.0.0.1:8332/

> curl --user myusername --data-binary '`"jsonrpc": "1.0", "id":"curltest", "method": "disconnectnode", "params": ¡"", 1¿ ñ' -H 'content-type: text/plain;' http://127.0.0.1:8332/

**getaddednodeinfo** ( "node" )

Returns information about the given added node, or all added nodes

(note that onetry addnodes are not listed here)

Arguments:

1. "node" (string, optional) If provided, return information about this specific node, otherwise all nodes are returned.

Result:

¡

`

"addednode" : "192.168.0.201", (string) The node IP address or name (as provided to addnode)

"connected" : true°false, (boolean) If connected

"addresses" : ¡ (list of objects) Only when connected = true

`

"address" : "192.168.0.201:8333", (string) The bitcoin server IP and port we're connected to

"connected" : "outbound" (string) connection, inbound or outbound

ñ

¿

ñ

,...

¿

Examples:

> bitcoin-cli getaddednodeinfo "192.168.0.201"

> curl --user myusername --data-binary '`"jsonrpc": "1.0", "id":"curltest", "method": "getaddednodeinfo", "params": ¡"192.168.0.201"¿ ñ' -H 'content-type: text/plain;' http://127.0.0.1:8332/

**getconnectioncount**

**ubuntu§philminer1**:**ç/.bitcoin**$ bitcoin-cli getconnectioncount

81

**getnettotals**

Returns information about network traffic, including bytes in, bytes out,

and current time.

Result:

`

"totalbytesrecv": n, (numeric) Total bytes received

"totalbytessent": n, (numeric) Total bytes sent

"timemillis": t, (numeric) Current UNIX time in milliseconds

"uploadtarget":

`

"timeframe": n, (numeric) Length of the measuring timeframe in seconds

"target": n, (numeric) Target in bytes

"target\_reached": true°false, (boolean) True if target is reached

"serve\_historical\_blocks": true°false, (boolean) True if serving historical blocks

"bytes\_left\_in\_cycle": t, (numeric) Bytes left in current time cycle

"time\_left\_in\_cycle": t (numeric) Seconds left in current time cycle

ñ

ñ

Examples:

> bitcoin-cli getnettotals

> curl --user myusername --data-binary '`"jsonrpc": "1.0", "id":"curltest", "method": "getnettotals", "params": ¡¿ ñ' -H 'content-type: text/plain;' http://127.0.0.1:8332/

**ubuntu§philminer1**:**ç/.bitcoin**$ bitcoin-cli getnettotals

`

"totalbytesrecv": 41955889788,

"totalbytessent": 1643638466387,

"timemillis": 1512955701277,

"uploadtarget": `

"timeframe": 86400,

"target": 0,

"target\_reached": false,

"serve\_historical\_blocks": true,

"bytes\_left\_in\_cycle": 0,

"time\_left\_in\_cycle": 0

ñ

**getnetworkinfo**

Returns an object containing various state info regarding P2P networking.

Result:

`

"version": xxxxx, (numeric) the server version

"subversion": "/Satoshi:x.x.x/", (string) the server subversion string

"protocolversion": xxxxx, (numeric) the protocol version

"localservices": "xxxxxxxxxxxxxxxx", (string) the services we offer to the network

"localrelay": true°false, (bool) true if transaction relay is requested from peers

"timeoffset": xxxxx, (numeric) the time offset

"connections": xxxxx, (numeric) the number of connections

"networkactive": true°false, (bool) whether p2p networking is enabled

"networks": ¡ (array) information per network

`

"name": "xxx", (string) network (ipv4, ipv6 or onion)

"limited": true°false, (boolean) is the network limited using -onlynet?

"reachable": true°false, (boolean) is the network reachable?

"proxy": "host:port" (string) the proxy that is used for this network, or empty if none

"proxy\_randomize\_credentials": true°false, (string) Whether randomized credentials are used

ñ

,...

¿,

"relayfee": x.xxxxxxxx, (numeric) minimum relay fee for transactions in BTC/kB

"incrementalfee": x.xxxxxxxx, (numeric) minimum fee increment for mempool limiting or BIP 125 replacement in BTC/kB

"localaddresses": ¡ (array) list of local addresses

`

"address": "xxxx", (string) network address

"port": xxx, (numeric) network port

"score": xxx (numeric) relative score

ñ

,...

¿

"warnings": "..." (string) any network warnings

ñ

Examples:

> bitcoin-cli getnetworkinfo

> curl --user myusername --data-binary '`"jsonrpc": "1.0", "id":"curltest", "method": "getnetworkinfo", "params": ¡¿ ñ' -H 'content-type: text/plain;' http://127.0.0.1:8332/

**ubuntu§philminer1**:**ç/.bitcoin**$ bitcoin-cli getnetworkinfo

`

"version": 150100,

"subversion": "/Satoshi:0.15.1/",

"protocolversion": 70015,

"localservices": "000000000000000d",

"localrelay": true,

"timeoffset": -1,

"networkactive": true,

"connections": 81,

"networks": ¡

`

"name": "ipv4",

"limited": false,

"reachable": true,

"proxy": "",

"proxy\_randomize\_credentials": false

ñ,

`

"name": "ipv6",

"limited": false,

"reachable": true,

"proxy": "",

"proxy\_randomize\_credentials": false

ñ,

`

"name": "onion",

"limited": true,

"reachable": false,

"proxy": "",

"proxy\_randomize\_credentials": false

ñ

¿,

"relayfee": 0.00001000,

"incrementalfee": 0.00001000,

"localaddresses": ¡

`

"address": "154.56.42.45",

"port": 8333,

"score": 45866

ñ

¿,

"warnings": ""

**getpeerinfo**

**ubuntu§philminer1**:**ç/.bitcoin**$ bitcoin-cli getpeerinfo

¡

`

"id": 2,

"addr": "184.61.213.5:8333",

"addrlocal": "154.56.42.45:47444",

"addrbind": "154.56.42.45:47444",

"services": "000000000000000d",

"relaytxes": true,

"lastsend": 1512955833,

"lastrecv": 1512955832,

"bytessent": 222427030,

"bytesrecv": 5243592902,

"conntime": 1511665005,

"timeoffset": 0,

"pingtime": 0.145009,

"minping": 0.034936,

"version": 70015,

"subver": "/Satoshi:0.15.1/",

"inbound": false,

"addnode": false,

"startingheight": 496130,

"banscore": 0,

"synced\_headers": 498653,

"synced\_blocks": 498653,

"inflight": ¡

¿,

"whitelisted": false,

"bytessent\_per\_msg": `

"addr": 464145,

"blocktxn": 37642,

"cmpctblock": 3958865,

"feefilter": 8768,

"getaddr": 24,

"getblocktxn": 7324,

"getdata": 20079946,

"getheaders": 1021,

"headers": 127705,

"inv": 149786795,

"notfound": 424367,

"ping": 344096,

"pong": 343584,

"reject": 14101,

"sendcmpct": 528,

"sendheaders": 24,

"tx": 46827945,

"verack": 24,

"version": 126

ñ,

"bytesrecv\_per\_msg": `

"addr": 546947,

"block": 4935108975,

"blocktxn": 806261,

"cmpctblock": 12954449,

"feefilter": 9056,

"getblocktxn": 1037,

"getdata": 5020697,

"getheaders": 1021,

"headers": 42587,

"inv": 79002341,

"notfound": 29856,

"ping": 343584,

"pong": 344096,

"reject": 65505,

"sendcmpct": 1518,

"sendheaders": 24,

"tx": 209314798,

"verack": 24,

"version": 126

ñ

ñ,

`

"id": 8,

"addr": "5.188.54.122:8333",

"addrlocal": "154.56.42.45:47376",

"addrbind": "154.56.42.45:47376",

"services": "000000000000000d",

"relaytxes": true,

"lastsend": 1512955833,

"lastrecv": 1512955827,

"bytessent": 246099965,

"bytesrecv": 5875869922,

"conntime": 1511665025,

"timeoffset": 65,

"pingtime": 0.16471,

"minping": 0.152752,

"version": 70014,

"subver": "/Satoshi:0.13.1/",

"inbound": false,

"addnode": false,

"startingheight": 496130,

"banscore": 0,

"synced\_headers": 498653,

"synced\_blocks": 498653,

"inflight": ¡

¿,

"whitelisted": false,

"bytessent\_per\_msg": `

"addr": 457905,

"block": 1037076,

"blocktxn": 65319,

"cmpctblock": 27638320,

"feefilter": 10112,

"getaddr": 24,

"getblocktxn": 1286,

"getdata": 18187311,

"getheaders": 164381,

"headers": 3604,

"inv": 151181359,

"notfound": 237953,

"ping": 344000,

"pong": 344096,

"reject": 34952,

"sendcmpct": 1518,

"sendheaders": 24,

"tx": 46390575,

"verack": 24,

"version": 126

ñ,

"bytesrecv\_per\_msg": `

"addr": 372917,

"block": 5544898450,

"blocktxn": 123446,

"cmpctblock": 4102349,

"feefilter": 9568,

"getblocktxn": 491,

"getdata": 4327661,

"getheaders": 1021,

"headers": 178673,

"inv": 86617928,

"notfound": 65177,

"ping": 344096,

"pong": 344000,

"reject": 112444,

"sendcmpct": 99,

"sendheaders": 24,

"tx": 234371428,

"verack": 24,

"version": 126

ñ

ñ,

`

"id": 313,

"addr": "54.191.174.46:37011",

"addrbind": "154.56.42.45:8333",

"services": "0000000000000000",

"relaytxes": false,

"lastsend": 1512955812,

"lastrecv": 1512955813,

"bytessent": 385434371,

"bytesrecv": 404892,

"conntime": 1511677101,

"timeoffset": -1097,

"pingtime": 0.073768,

"minping": 0.07239900000000001,

"version": 70015,

"subver": "/ViaBTC:bitpeer.0.2.0/",

"inbound": true,

"addnode": false,

"startingheight": 0,

"banscore": 0,

"synced\_headers": 498629,

"synced\_blocks": -1,

"inflight": ¡

¿,

"whitelisted": false,

"bytessent\_per\_msg": `

"addr": 472975,

"block": 384348278,

"feefilter": 9280,

"headers": 262487,

"inv": 183,

"ping": 340928,

"sendcmpct": 66,

"sendheaders": 24,

"verack": 24,

"version": 126

ñ,

"bytesrecv\_per\_msg": `

"getdata": 24034,

"headers": 39750,

"pong": 340928,

"sendheaders": 24,

"verack": 24,

"version": 132

ñ

ñ,

`

**listbanned** --- list all banned IP addresses/subnets

**ping**

Requests that a ping be sent to all other nodes, to measure ping time.

Results provided in getpeerinfo, pingtime and pingwait fields are decimal seconds.

Ping command is handled in queue with all other commands, so it measures processing backlog, not just network ping.

**setban** "subnet" "add|remove" (bantime) (absolute)

Attempts to add or remove an IP/Subnet from the banned list.

Arguments:

1. "subnet" (string, required) The IP/Subnet (see getpeerinfo for nodes IP) with an optional netmask (default is /32 = single IP)

2. "command" (string, required) 'add' to add an IP/Subnet to the list, 'remove' to remove an IP/Subnet from the list

3. "bantime" (numeric, optional) time in seconds how long (or until when if ¡absolute¿ is set) the IP is banned (0 or empty means using the default time of 24h which can also be overwritten by the -bantime startup argument)

4. "absolute" (boolean, optional) If set, the bantime must be an absolute timestamp in seconds since epoch (Jan 1 1970 GMT)

Examples:

> bitcoin-cli setban "192.168.0.6" "add" 86400

> bitcoin-cli setban "192.168.0.0/24" "add"

> curl --user myusername --data-binary '`"jsonrpc": "1.0", "id":"curltest", "method": "setban", "params": ¡"192.168.0.6", "add", 86400¿ ñ' -H 'content-type: text/plain;' http://127.0.0.1:8332/

**setnetworkactive** true|false

Disable/enable all p2p network activity.

Arguments:

1. "state" (boolean, required) true to enable networking, false to disable

**ubuntu§philminer1**:**ç/.bitcoin**$ bitcoin-cli setnetworkactive

error code: -1

error message:

setnetworkactive true°false

Disable/enable all p2p network activity.

Arguments:

1. "state" (boolean, required) true to enable networking, false to disable

**== Rawtransactions ==**

**combinerawtransaction** ["hexstring",...]

Combine multiple partially signed transactions into one transaction.

The combined transaction may be another partially signed transaction or a

fully signed transaction.

Arguments:

1. "txs" (string) A json array of hex strings of partially signed transactions

¡

"hexstring" (string) A transaction hash

,...

¿

Result:

"hex" (string) The hex-encoded raw transaction with signature(s)

Examples:

> bitcoin-cli combinerawtransaction ¡"myhex1", "myhex2", "myhex3"¿

**createrawtransaction** [{"txid":"id","vout":n},...] {"address":amount,"data":"hex",...} ( locktime ) ( replaceable )

Create a transaction spending the given inputs and creating new outputs.

Outputs can be addresses or data.

Returns hex-encoded raw transaction.

Note that the transaction's inputs are not signed, and

it is not stored in the wallet or transmitted to the network.

Arguments:

1. "inputs" (array, required) A json array of json objects

¡

`

"txid":"id", (string, required) The transaction id

"vout":n, (numeric, required) The output number

"sequence":n (numeric, optional) The sequence number

ñ

,...

¿

2. "outputs" (object, required) a json object with outputs

`

"address": x.xxx, (numeric or string, required) The key is the bitcoin address, the numeric value (can be string) is the BTC amount

"data": "hex" (string, required) The key is "data", the value is hex encoded data

,...

ñ

3. locktime (numeric, optional, default=0) Raw locktime. Non-0 value also locktime-activates inputs

4. replaceable (boolean, optional, default=false) Marks this transaction as BIP125 replaceable.

Allows this transaction to be replaced by a transaction with higher fees. If provided, it is an error if explicit sequence numbers are incompatible.

Result:

"transaction" (string) hex string of the transaction

Examples:

> bitcoin-cli createrawtransaction "¡`Ñ"txidÑ":Ñ"myidÑ",Ñ"voutÑ":0ñ¿" "`Ñ"addressÑ":0.01ñ"

> bitcoin-cli createrawtransaction "¡`Ñ"txidÑ":Ñ"myidÑ",Ñ"voutÑ":0ñ¿" "`Ñ"dataÑ":Ñ"00010203Ñ"ñ"

> curl --user myusername --data-binary '`"jsonrpc": "1.0", "id":"curltest", "method": "createrawtransaction", "params": ¡"¡`Ñ"txidÑ":Ñ"myidÑ",Ñ"voutÑ":0ñ¿", "`Ñ"addressÑ":0.01ñ"¿ ñ' -H 'content-type: text/plain;' http://127.0.0.1:8332/

> curl --user myusername --data-binary '`"jsonrpc": "1.0", "id":"curltest", "method": "createrawtransaction", "params": ¡"¡`Ñ"txidÑ":Ñ"myidÑ",Ñ"voutÑ":0ñ¿", "`Ñ"dataÑ":Ñ"00010203Ñ"ñ"¿ ñ' -H 'content-type: text/plain;' http://127.0.0.1:8332/

**decoderawtransaction** "hexstring"

Return a JSON object representing the serialized, hex-encoded transaction.

Arguments:

1. "hexstring" (string, required) The transaction hex string

Result:

`

"txid" : "id", (string) The transaction id

"hash" : "id", (string) The transaction hash (differs from txid for witness transactions)

"size" : n, (numeric) The transaction size

"vsize" : n, (numeric) The virtual transaction size (differs from size for witness transactions)

"version" : n, (numeric) The version

"locktime" : ttt, (numeric) The lock time

"vin" : ¡ (array of json objects)

`

"txid": "id", (string) The transaction id

"vout": n, (numeric) The output number

"scriptSig": ` (json object) The script

"asm": "asm", (string) asm

"hex": "hex" (string) hex

ñ,

"txinwitness": ¡"hex", ...¿ (array of string) hex-encoded witness data (if any)

"sequence": n (numeric) The script sequence number

ñ

,...

¿,

"vout" : ¡ (array of json objects)

`

"value" : x.xxx, (numeric) The value in BTC

"n" : n, (numeric) index

"scriptPubKey" : ` (json object)

"asm" : "asm", (string) the asm

"hex" : "hex", (string) the hex

"reqSigs" : n, (numeric) The required sigs

"type" : "pubkeyhash", (string) The type, eg 'pubkeyhash'

"addresses" : ¡ (json array of string)

"12tvKAXCxZjSmdNbao16dKXC8tRWfcF5oc" (string) bitcoin address

,...

¿

ñ

ñ

,...

¿,

ñ

Examples:

> bitcoin-cli decoderawtransaction "hexstring"

> curl --user myusername --data-binary '`"jsonrpc": "1.0", "id":"curltest", "method": "decoderawtransaction", "params": ¡"hexstring"¿ ñ' -H 'content-type: text/plain;' http://127.0.0.1:8332/

**decodescript** "hexstring"

Decode a hex-encoded script.

Arguments:

1. "hexstring" (string) the hex encoded script

Result:

`

"asm":"asm", (string) Script public key

"hex":"hex", (string) hex encoded public key

"type":"type", (string) The output type

"reqSigs": n, (numeric) The required signatures

"addresses": ¡ (json array of string)

"address" (string) bitcoin address

,...

¿,

"p2sh","address" (string) address of P2SH script wrapping this redeem script (not returned if the script is already a P2SH).

ñ

Examples:

> bitcoin-cli decodescript "hexstring"

> curl --user myusername --data-binary '`"jsonrpc": "1.0", "id":"curltest", "method": "decodescript", "params": ¡"hexstring"¿ ñ' -H 'content-type: text/plain;' http://127.0.0.1:8332/

**fundrawtransaction** "hexstring" ( options )

Add inputs to a transaction until it has enough in value to meet its out value. This will not modify existing inputs, and will add at most one change output to the outputs. No existing outputs will be modified unless "subtractFeeFromOutputs" is specified.

Note that inputs which were signed may need to be resigned after completion since in/outputs have been added. The inputs added will not be signed, use signrawtransaction for that.

Note that all existing inputs must have their previous output transaction be in the wallet.

Note that all inputs selected must be of standard form and P2SH scripts must be in the wallet using importaddress or addmultisigaddress (to calculate fees). You can see whether this is the case by checking the "solvable" field in the listunspent output. Only pay-to-pubkey, multisig, and P2SH versions thereof are currently supported for watch-only

Arguments:

1. "hexstring" (string, required) The hex string of the raw transaction

2. options (object, optional)

`

"changeAddress" (string, optional, default pool address) The bitcoin address to receive the change

"changePosition" (numeric, optional, default random) The index of the change output

"includeWatching" (boolean, optional, default false) Also select inputs which are watch only

"lockUnspents" (boolean, optional, default false) Lock selected unspent outputs

"feeRate" (numeric, optional, default not set: makes wallet determine the fee) Set a specific feerate (BTC per KB)

"subtractFeeFromOutputs" (array, optional) A json array of integers.

The fee will be equally deducted from the amount of each specified output.

The outputs are specified by their zero-based index, before any change output is added.

Those recipients will receive less bitcoins than you enter in their corresponding amount field.

If no outputs are specified here, the sender pays the fee.

¡vout\_index,...¿

"replaceable" (boolean, optional) Marks this transaction as BIP125 replaceable.

Allows this transaction to be replaced by a transaction with higher fees

"conf\_target" (numeric, optional) Confirmation target (in blocks)

"estimate\_mode" (string, optional, default=UNSET) The fee estimate mode, must be one of:

"UNSET"

"ECONOMICAL"

"CONSERVATIVE"

ñ

for backward compatibility: passing in a true instead of an object will result in `"includeWatching":trueñ

Result:

`

"hex": "value", (string) The resulting raw transaction (hex-encoded string)

"fee": n, (numeric) Fee in BTC the resulting transaction pays

"changepos": n (numeric) The position of the added change output, or -1

ñ

Examples:

Create a transaction with no inputs

> bitcoin-cli createrawtransaction "¡¿" "`Ñ"myaddressÑ":0.01ñ"

Add sufficient unsigned inputs to meet the output value

> bitcoin-cli fundrawtransaction "rawtransactionhex"

Sign the transaction

> bitcoin-cli signrawtransaction "fundedtransactionhex"

Send the transaction

> bitcoin-cli sendrawtransaction "signedtransactionhex"

**getrawtransaction** "txid" ( verbose )

**sendrawtransaction** "hexstring" ( allowhighfees )

**signrawtransaction** "hexstring" ( [{"txid":"id","vout":n,"scriptPubKey":"hex","redeemScript":"hex"},...] ["privatekey1",...] sighashtype )

**== Util ==**

createmultisig nrequired ["key",...]

estimatefee nblocks

estimatesmartfee conf\_target ("estimate\_mode")

signmessagewithprivkey "privkey" "message"

validateaddress "address"

verifymessage "address" "signature" "message"

**== Wallet == Not Included for Security**

abandontransaction "txid"

abortrescan

addmultisigaddress nrequired ["key",...] ( "account" )

addwitnessaddress "address"

backupwallet "destination"

bumpfee "txid" ( options )

dumpprivkey "address"

dumpwallet "filename"

getaccount "address"

getaccountaddress "account"

getaddressesbyaccount "account"

getbalance ( "account" minconf include\_watchonly )

getnewaddress ( "account" )

getrawchangeaddress

getreceivedbyaccount "account" ( minconf )

getreceivedbyaddress "address" ( minconf )

gettransaction "txid" ( include\_watchonly )

getunconfirmedbalance

getwalletinfo

importaddress "address" ( "label" rescan p2sh )

importmulti "requests" ( "options" )

importprivkey "privkey" ( "label" ) ( rescan )

importprunedfunds

importpubkey "pubkey" ( "label" rescan )

importwallet "filename"

keypoolrefill ( newsize )

listaccounts ( minconf include\_watchonly)

listaddressgroupings

listlockunspent

listreceivedbyaccount ( minconf include\_empty include\_watchonly)

listreceivedbyaddress ( minconf include\_empty include\_watchonly)

listsinceblock ( "blockhash" target\_confirmations include\_watchonly include\_removed )

listtransactions ( "account" count skip include\_watchonly)

listunspent ( minconf maxconf ["addresses",...] [include\_unsafe] [query\_options])

listwallets

lockunspent unlock ([{"txid":"txid","vout":n},...])

move "fromaccount" "toaccount" amount ( minconf "comment" )

removeprunedfunds "txid"

sendfrom "fromaccount" "toaddress" amount ( minconf "comment" "comment\_to" )

sendmany "fromaccount" {"address":amount,...} ( minconf "comment" ["address",...] replaceable conf\_target "estimate\_mode")

sendtoaddress "address" amount ( "comment" "comment\_to" subtractfeefromamount replaceable conf\_target "estimate\_mode")

setaccount "address" "account"

settxfee amount

signmessage "address" "message"

walletlock

walletpassphrase "passphrase" timeout

walletpassphrasechange "oldpassphrase" "newpassphrase"