|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Command** | **Parameters NOTE:**  **“< >” is required;**  **“[ ]” is optional.** | **Description** | **Requires unlocked wallet?** | **Return result** |
| addnode | <node> <add/remove/onetry> | Attempts add or remove <node> from the addnode list or try a connection to <node> once. | N | Result:  Error:null  id:null  result:null |
| backupwallet | <destination> | Safely copies wallet.dat to destination, which can be a directory or a path with filename. | N | Result:  Error:null  id:null  result:null |
| createcontracttx | <userregid><appid><amount><contract><fee>[height ,default = the tip block height] | create contract transaction | Y | Result:  hash : (string)the hash of contract tx; |
| createcontracttxraw | <height><fee><amount><address><contract> | Create contract transaction from hex string | N | Result:  rawtx: (string)the hash of contract tx; |
| dispersebalance | <send address><amount> | coins [scattered](javascript:void(0);) [to](javascript:void(0);) [the](javascript:void(0);) [address](javascript:void(0);) [of](javascript:void(0);) [his](javascript:void(0);) [wallet](javascript:void(0);).  <amount>other addresses received | Y | Result:  Tx:array  [  (string)the hash of tx;  ] |
| dropprivkey |  | drop private key from wallet | Y | Result:  info:(string)message if success message is "wallet is ready for cool miner" |
| dumpwallet | <filename> | Dumps all wallet keys in a human-readable format.And write to <filename> | Y | Result:  Info:(string)message if success message is "dump ok";  key size:( numeric)the size of key; |
| dumpprivkey | <dacrsaddress> | Reveals the private key corresponding to <dacrsaddress> | Y | Result:  minerkey:(string)  privkey:(string) |
| encryptwallet | <passphrase> | Encrypts the wallet with <passphrase>. | N | Result:  encrypt:(boolean)true if encrytwallet execute successfully |
| generateblock | <address> | cteate a block with the appointed address | N | Result:  blockhash:the hash of block |
| getaccountinfo | <address> | Returns the account information with the given address. | N | Result  Address:(string)the address in wallet  KeyID:(string) KeyID the account  RegID:(string)the ID of address which was activated;  PublicKey:(string) public key of the account;  MinerPKey:(string) public key of the account for miner;  Balance:(uint64\_t) total money;  CoinDays:(uint64\_t) coin day  UpdateHeight:(int) update height  CurCoinDays:(uint64\_t)  postion:(string) |
| getaddednodeinfo | <dns> [node] | Returns information about the given added node, or all added nodes  (note that onetry addnodes are not listed here) If dns is false, only a list of added nodes will be provided, otherwise connected information will also be available. | N | Result:  If dns if false, a list of added node:  [addednode:(string)the node ip address;  ...]  Else information about the specific node:  [addednode:(string)The node ip address;  connected:true|false,(boolean);  addresses:  [  address: (string) The Dacrs server host and port  connected:(string) connection, inbound or outbound  ...  ]  ...] |
| getappinfo | <scriptid> | Get app info | N | Result:  scriptId:(string)The script ID;  scriptId2:(string)  description:(string)  scriptContent:(string) |
| getappaccinfo | <scriptid><address> | get appaccount info | N | Result:  mAccuserID:(string)  FreeValues:(uint64\_t)  vFreezeFund:a list of freezed fund  [value:(uint64\_t);  nHeight:(int);  vTag: (string)vTag of the tx which create the fund;  ...] |
| getappkeyvalue | <scriptid><array> | get application key value | N | Result:  key:(string)  value:( numeric)  confirmHeight:( numeric)  confirmedtime:( numeric) |
| getassets | <scriptid> | The collection of all assets | N | Result:  TotalAssets: uint64\_t  Lists:array  [  Address:string  FreeValues: uint64\_t  FreezedFund: uint64\_t  ] |
| getbalance | [account] [minconf=1] | If [account] is not specified, returns the server's total available balance. If [account] is specified, returns the balance in the account.  If [minconf] is 1; Only include transactions confirmed. Default max value is 30, can configure -maxconf parameter changer the max value. | N | Result:Success  balance:( double) |
| getbestblockhash |  | Returns the hash of the best (tip) block in the longest block chain. | N | Result:  (string)the block hash hex encoded; |
| getblock | <hash or index>[verbose] | Returns information about the block with the given hash or index.If verbose is true,return a json object, false return the hex encoded data | N | Result:(for verbose = true)  hash:(string)the block hash(same as provided);  confirmations: (numeric) The number of confirmations;  size: (numeric) The block size;  height: (numeric) The block height or index;  version: (numeric) The block version;  merkleroot: (string) The merkle root;  txnumber:( numeric);  tx: (array of string) The transaction ids  [  transactionid: (string) The transaction id  ...  ];  time: (numeric) The block time in seconds since epoch (Jan 1 1970 GMT);  nonce: (numeric) The nonce;  bits: (string) The bits  difficulty: (numeric) The difficulty;  chainwork:(string)  fuel:(int64\_t)  fuelrate:(int)  previousblockhash: (string) The hash of the previous block;  nextblockhash: (string) The hash of the next block;  Result:(for verbose = false):  data: (string) A string that is serialized, hex-encoded data for block 'hash'; |
| getblockcount |  | Returns the number of blocks in the longest block chain. | N | Result:  (numeric)The current block count; |
| getblockchaininfo |  | Returns an object containing various state info regarding block chain processing. | N | Result:  chain: (string) current chain (main, testnet3, regtest);  blocks: (numeric) the current number of blocks processed in the server;  bestblockhash: (string) the hash of the currently best block;  difficulty: (numeric) the current difficulty;  verificationprogress: (numeric) estimate of verification progress [0..1];  chainwork: (string) total amount of work in active chain, in hexadecimal; |
| getblockhash | <index> | Returns hash of block in best-block-chain at <index>; index 0 is the [genesis block](https://en.bitcoin.it/wiki/Genesis_block) | N | Result:  hash: (string) The block hash; |
| getnettotals |  | Returns information about network traffic, including bytes in, bytes out | N | Result:  Totalbytesrecv: (numeric) Total bytes received;  Totalbytessent: (numeric) Total bytes sent;  Timemillis: (numeric) Total cpu time; |
| getconnectioncount |  | Returns the number of connections to other nodes. | N | Result:  (numeric) The connection count |
| getdacrsstate | <num> | Returns state data about the recently num blocks. | N | Result:  blocktime: (numeric)array,get the time of each block;  difficulty: (numeric)array,get the difficulty of each block;  transactions: (numeric)array,get the transactions of each block;  fuel: (numeric)array,get fuel of each block;  blockminer: (numeric)array,get the miner of each block; |
| getdifficulty |  | Returns the proof-of-work difficulty as a multiple of the minimum difficulty. | N | Result:  (numeric) the proof-of-work difficulty as a multiple of the minimum difficulty; |
| getinfo |  | Returns an object containing various state info. | N | Result:  version: (numeric) the server version;  fullversion: (string) the server fullversion;  protocolversion: (numeric) the protocol version;  walletversion: (numeric) the wallet version;  balance: (numeric) the total Dacrs balance of the wallet;  blocks: (numeric) the current number of blocks processed in the server;  timeoffset: (numeric) the time offset;  connections: (numeric) the number of connections;  proxy: (string, optional) the proxy used by the server;  difficulty: (numeric) the current difficulty;  nettype: (string) the net type;  chainwork: (string) the chainwork of the tip block in chainActive;  tipblocktime: (numeric) the nTime of the tip block in chainActive;  unlocked\_until: (numeric) the timestamp in seconds since epoch (midnight Jan 1 1970 GMT) that the wallet is unlocked for transfers, or 0 if the wallet is locked;  paytxfee: (numeric) the transaction fee set in btc/kb;  relayfee: (numeric) minimum relay fee for non-free transactions in btc/kb;  fuelrate: (numeric) the fuelrate of the tip block in chainActive;  fuel: (numeric) the fuel of the tip block in chainActive;  data directory: (string) the data directory;  tip block hash: (string) the tip block hash;  errors: (string) any error messages; |
| getmininginfo |  | Returns an object containing mining-related information:   * blocks * currentblocksize * currentblocktx * difficulty * errors * generate * genproclimit * hashespersec * pooledtx * testnet | N | Result:  blocks: (numeric) The current block;  currentblocksize: (numeric) The last block size;  currentblocktx: (numeric) The last block transaction;  difficulty: (numeric) The current difficulty;  errors: (string) Current errors;  genproclimit: (numeric) The processor limit for generation. -1 if no generation. (see getgenerate or setgenerate calls);  networkhashps: (numeric) Hashes per second estimated  pooledtx: (numeric) The size of the mem pool;  nettype: (string)TESTNET If using testnet, MAIN if using main,REGTEST if using regtest;  posmaxnonce:(numeric)  generate: (boolean) If the generation is on or off (see getgenerate or setgenerate calls); |
| getnewaddress | [isminer] | Returns a new address for receiving payments. If [isminer] is ture will create a miner key,otherwise will only return a new address. | Y | Result:  addr:(string)new address;  minerpubkey:(string) If [isminer] is ture will create a miner key,otherwise will only return a new address. |
| getnetworkhashps | [blocks][height] | Returns the estimated network hashes per second based on the last n blocks.   1. blocks (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. | N | Result:  (numeric) Hashes per second estimated; |
| getnetworkinfo |  | Returns an object containing various state info regarding P2P network. | N | Result:  version: (numeric) the server version;  protocolversion: (numeric) the protocol version;  timeoffset: (numeric) the time offset;  connections: (numeric) the number of connections;  proxy: (string, optional) the proxy used by the server;  relayfee: (numeric) minimum relay fee for non-free transactions in btc/kb;  localaddresses: (array) list of local [addresses;  address: (string) network address;  port: (numeric) network port;  score: (numeric) relative score;  ] |
| getpeerinfo |  | Returns data about each connected node. | N | Result:  addr:(string) The ip address and port of the peer;  addrlocal:(string) local address; services:(string) The services;  lastsend:(numeric) The time in seconds since epoch (Jan 1 1970 GMT) of the last send;  lastrecv:(numeric) The time in seconds since epoch (Jan 1 1970 GMT) of the last receive;  bytessent:(numeric) The total bytes sent; bytesrecv:(numeric) The total bytes received;  conntime:(numeric) The connection time in seconds since epoch (Jan 1 1970 GMT)  pingtime:(numeric) ping time;  pingwait:(numeric) ping wait;  version:(numeric) The peer version, such as 7001;  subver:(string) The string version;  inbound: true|false, (boolean) Inbound (true) or Outbound (false);  startingheight:(numeric) The starting height (block) of the peer;  banscore:(numeric) The ban score (stats.nMisbehavior)  syncnode : true|false (booleamn) if sync node; |
| getrawmempool | [verbose] | Returns all transaction ids in memory pool.If verbose is true,return a json object, false return array of transaction ids. | N | Result:(for verbose = false)  [  transactionid: (string) The transaction id;  ...  ] (json array of string);  Result:(for verbose = true)  [  transactionid: (json object) {  size:(numeric) transaction size in bytes;  fee:(numeric) transaction fee in Dacrss;  time: (numeric) local time transaction entered pool in seconds since 1 Jan 1970 GMT  height:(numeric) block height when transaction entered pool; startingpriority:(numeric) priority when transaction entered pool;  currentpriority:(numeric) transaction priority now;  depends: [ (array) unconfirmed transactions used as inputs for this transaction;  transactionid:(string) parent transaction id  ...  ]  }  ] |
| getscriptdata | <scriptid><pagsize or key><index> | get the script data by given scripted.  < scriptid ><key> or  < scriptid >< pagsize ><index> | N | Result:array  [key:(string)  value:(string)  ...] |
| getscriptvalidedata | <scriptid><pagsize><index> | get script valide data. | N | Result:  [  key:(string)  value:(string)  ...  ] |
| gettotalassets | <scriptid> | Returns the total of assets | N | Result:  TotalAssets: uint64\_t |
| gettxdetail | <txhash> | Returns an object about the transaction detail information by <txhash> | N | Result:  hash:(string)  txtype:(string)  ver:( numeric)  addr:(string)  desregid:(string)  desaddr:(string)  money:( numeric)  fees:( numeric)  height:( numeric)  Contract:(string)  blockhash:(string)  confirmHeight:( numeric)  confirmedtime:( numeric)  rawtx:(string) |
| getwalletinfo |  | Returns an object containing various wallet state info. | N | Result:  walletversion:(numeric) the wallet version;  balance:(numeric) the total Dacrs balance of the wallet; Inblocktx:(numeric) the size of transactions in the wallet; uncomfirmedtx:(numeric) the size of unconfirmtx transactions in the wallet;  unlocked\_until:(numeric) the timestamp in seconds since epoch (midnight Jan 1 1970 GMT) that the wallet is unlocked for transfers, or 0 if the wallet is locked; |
| help | [command] | List commands, or get help for a command. | N | Result:  (string) The Help Text; |
| importprivkey | <dacrsprivkey> [label] [rescan=true] | Adds a private key (as returned by dumpprivkey) to your wallet. This may take a while, as a [rescan](https://en.bitcoin.it/wiki/How_to_import_private_keys#Import_Private_key.28s.29) is done, looking for existing transactions. Note: There's no need to import public key, as in [ECDSA](https://en.bitcoin.it/wiki/Elliptic_Curve_Digital_Signature_Algorithm) (unlike RSA) this can be computed from private key. | Y | Result:  imorpt key address:(string)the address of which import the key; |
| importwallet | <filename> | Import keys from a wallet dump file (see dumpwallet). | Y | Result:  imorpt key size:(numeric)the size of import key; |
| islocked |  | Return an object about the wallet is lock or unlock. | N | Result:  islock:( numeric)0:decrypted,1:encryped and unlocked,encryped and locked; |
| listaddr |  | return Array containing address,balance,haveminerkey,regid information. | N | Result:array  [  addr:(string)addr in wallet;  balance:( numeric)balance in current addr;  haveminerkey:(boolean);  regid:(string)  ..  ] |
| listapp | <showDetail> | get the list register script.  1. showDetail (boolean, required)true to show scriptContent,otherwise to not show it. | N | Result:  [  scriptId:(string)the id of script;  scriptId2:(string)  description:(string)  ...  ] |
| listcheckpoint |  | Returns the list of checkpoint | N | Result:  The array of checkpoint include (height and hash); |
| listtx |  | get all confirm transactions and all unconfirm transactions from wallet. | N | Result:  ConfirmTx:array  [  tx:(string)hash of tx;  ],  UnConfirmTx:array  [  tx:(string)hash of unconfirmed tx;  ] |
| listtxcache |  | get all transactions in cahce | N | Result:  [  blockhash:(string)the hash of block;  txcache:(string)the hash of tx;  ...  ] |
| listunconfirmedtx |  | get the list of unconfirmedtx. | N | Result:  UnConfirmTx:  [  tx:(string)hash of unconfirmed tx;  ...  ] |
| notionalpoolingbalance | <receive address><amount> | Collects the currency to an address  <amount>other address whose amount is greater than this can be collected to receive address | Y | Result:  Tx:(string)the hash of tx |
| registaccounttx | <address><fee> | register secure account | Y | Result:  hash:(string)the hash of tx; |
| registaccounttxraw | <height><fee><publickey>[minerpublickey] | create a register account transaction | N | Result:  rawtx:(string);  signhash:(string); |
| registerapptx | <address><filepath><fee>[height][scriptdescription] | create a register script transaction | Y | Result:  hash:(string)the hash of tx; |
| registerscripttxraw | <height><fee><address><flag><script or scriptid>[script description] | Register script.   1. Height (numeric required) :valod height 2. Fee: (numeric required) pay to miner 3. address: (string required)for send 4. flag: (numeric, required) 0-1 5. script or scriptid: (string required), if flag=0 is script's file path, else if flag=1 scriptid 6. script description:(string optional) new script description. | N | Result:  rawtx:(string); |
| sendtoaddress | [dacrsaddress]<receive address><amount> | Send an amount to a given address. The amount is a real and is rounded to the nearest 0.00000001. Returns the transaction ID <txhash> if successful. | Y | Result:  hash: (string) The transaction id. |
| sendtoaddressraw | <height><fee><amount><srcaddress><recvaddress> | create normal transaction by hegiht,fee,amount,srcaddress, recvaddress. | N | Result:  rawtx: (string) .  signhash:(string); |
| sendtoaddresswithfee | [sendaddress]<recvaddress><amount><fee> | Send an amount to a given address with fee. The amount is a real and is rounded to the nearest 0.00000001.  Sendaddress is optional. | Y | Result:  hash: (string); |
| setgenerate | <generate> [genproclimit] | <generate> is true or false to turn generation on or off. Generation is limited to [genproclimit] processors, -1 is unlimited. | N | Result:  msg:(string)msg is "in mining" if generate is true; msg is "stoping mining" if generate is false; |
| settxfee | <amount> | <amount> is a real and is rounded to the nearest 0.00000001 | N | Result:  True|false (boolean) returns true if successful; |
| signmessage | <dacrsaddress> <message> | Sign a message with the private key of an address. | Y | Result:  Signature: (string) The signature of the message encoded in base 64; |
| sigstr | <transaction><address> | signature transaction | N | Result:  rawtx:(string)the hash of transaction; |
| stop |  | Stop Dacrs server. | N | Stop server |
| submitblock | <hexdata> [optional-params-obj] | Attempts to submit new block to network.  1. hexdata (string, required) the hex-encoded block data to submit | N | Result:  status:OK  hash:the hash of block;  Result:(if failed)  Status:rejected  reject code:  info:(string) |
| submittx | <transaction> | submit transaction | Y | Result:  hash:(string)the hash of transaction |
| verifymessage | <dacrsaddress> <signature> <message> | Verify a signed message. | N | Result:  true|false (boolean) If the signature is verified or not; |
| verifychain | [checklevel][numblocks] | Verifies blockchain database.   1. checklevel (numeric, optional, 0-4, default=3), How thorough the block verification is. 2. numblocks (numeric, optional, default=288, 0=all) The number of blocks to check. | N | Result:  true|false (boolean) Verified or not; |
| walletlock |  | Removes the wallet encryption key from memory, locking the wallet. After calling this method, you will need to call walletpassphrase again before being able to call any methods which require the wallet to be unlocked. | N | Result:  true if success;else Error: running with an unencrypted wallet, but walletlock was called. |
| walletpassphrase | <passphrase> <timeout> | Stores the wallet decryption key in memory for <timeout> seconds. | N | Result:  passphrase:(boolean) true if command is successful; |
| walletpassphrasechange | <oldpassphrase> <newpassphrase> | Changes the wallet passphrase from <oldpassphrase> to <newpassphrase>. | N | Result:  chgpwd:(boolean)true if command is successful; |
| ping |  | Requests that a ping be sent to all other nodes, to measure ping time. | N | Result:  provided in getpeerinfo, pingtime and pingwait fields are decimal seconds;  Ping command is handled in queue with all other commands, so it measures processing |
| validateaddress | <address> | check the address is valide | N | Result:  ret:(boolean)true if the address is valide; fale if the address is invalide |
| getalltxinfo | [nlimitCount] | if no input params,return all transaction in wallet include confirmed and unconfirmed,else return number fo nlimitCount transaction relate. | N | Result:  Confirmed:array of confirmed txs  [  hash:(string)the hash of tx;  txtype:(string)the type of tx;  ver:( numeric)  regid:(string)  addr:(string)  desregid:(string)  desaddr:(string)  money:( numeric)  fees:( numeric)  height:( numeric)  Contract:(string)  blockhash:( string)  confirmHeight:( numeric)  confirmedtime:( numeric)  rawtx:(string)  ...  ]  UnConfirmed:array of unconfirmed txs  [  hash:(string)  txtype:(string)  ver:( numeric)  regid:(string)  addr:(string)  desregid:(string)  desaddr:(string)  money:( numeric)  fees:( numeric)  height:( numeric)  Contract:(string)  rawtx:(string)  ...  ] |
|  |  |  |  |  |