

Package ‘rbch’

January 10, 2022

Version 0.1-1

Date 2022-01-08

Title Extraction and Analysis of Data from the Bitcoin Cash (BCH)
Blockchain

Author Rucknium [cre, aut, cph] (<<https://orcid.org/0000-0001-5999-8950>>),
Bernhard Pfaff [aut, cph] (Creator of rbtc, the package that rbch is
originally based on.)

Maintainer Rucknium <Rucknium@protonmail.com>

Description Issues RPC-JSON calls to 'bitcoind', the daemon of Bitcoin Cash (BCH), to
extract transaction data from the blockchain. BCH is a fork of Bitcoin that permits
a greater number of transactions per second. A BCH daemon is available under an MIT
license from the Bitcoin Unlimited website <<https://www.bitcoinunlimited.info>>.

License GPL-3

URL <https://github.com/Rucknium/rbch>,
<https://www.bitcoinunlimited.info>

BugReports <https://github.com/Rucknium/rbch/issues>

SystemRequirements BCHunlimited (>= 1.9.2)

Depends R (>= 3.4.0)

Imports gmp, httr, methods, openssl, rjson

NeedsCompilation no

RoxygenNote 7.1.1

Repository CRAN

Date/Publication 2022-01-10 18:42:47 UTC

R topics documented:

addnode	3
ANSRPC-class	4
base58CheckDecode	5
base58CheckEncode	5

BCHADR-class	6
bkfee	7
blockattime	8
blockstats	8
BTCADR-class	9
clearbanned	10
concatHex	10
conrpc	11
CONRPC-class	12
containsPoint	12
date2int	13
decodeHex	14
decoderawtransaction	15
decodescript	15
disconnectnode	16
ecoperators	17
ecparam	18
ECPARAM-class	19
EcparamOrNull-class	20
ecpoint	20
ECPOINT-class	21
getaddednodeinfo	22
getbestblockhash	23
getblock	23
getblockchaininfo	24
getblockcount	25
getblockhash	26
getblockheader	27
getchaintips	28
getchaintxstats	28
getconnectioncount	29
getdifficulty	30
gethelp	31
getinfo	31
getmempoolancestors	32
getmempooldescendants	33
getmempoolentry	34
getmempoolinfo	35
getnettotals	35
getnetworkinfo	36
getpeerinfo	37
getrawmempool	38
getrawtransaction	39
gettxout	40
gettxoutproof	41
gettxoutsetinfo	42
getwalletinfo	42
hash160	43

hash256	44
int2date	45
intMaxDay	46
intMinDay	47
intRangeDay	48
intRangePeriod	48
isNull	49
listbanned	50
NullOrCharacter-class	51
NullOrInteger-class	51
ping	51
pruneblockchain	52
PubHash2BchAdr	53
PubHash2BtcAdr	54
PubKey2PubHash	54
rpcpost	55
show	56
startbch	56
startbtc	57
stopbch	58
stopbtc	58
timeofblock	59
txfee	60
txids	60
txinids	61
txstats	62
utxoage	62
utxotype	63
utxovalue	64
validBchAdr	64
validBtcAdr	65
verifychain	66
verifytxoutproof	67
Index	68

 addnode

RPC-JSON API: addnode

Description

Attempts to add or remove a node from the addnode list. Or try a connection to a node once.

Usage

```
addnode(con, node, command = c("add", "remove", "onetry"))
```

Arguments

con	object of class CONRPC.
node	character the node (see <code>getpeerinfo()</code> for nodes).
command	character '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.

Value

A S4-object of class ANSRPC.

Author(s)

Bernhard Pfaff

References

<https://bitcoin.org/en/developer-reference#addnode>, <https://bitcoin.org/en/developer-reference#remote-procedure-calls-rpcs>

See Also

Other Network RPCs: [clearbanned\(\)](#), [disconnectnode\(\)](#), [getaddednodeinfo\(\)](#), [getconnectioncount\(\)](#), [getnettotals\(\)](#), [getnetworkinfo\(\)](#), [getpeerinfo\(\)](#), [listbanned\(\)](#), [ping\(\)](#)

ANSRPC-class

The ANSRPC class

Description

This class definition is employed to cast the JSON-objects returned by API-calls to bitcoind.

Slots

rpcname	character the name of the API.
result	ANY the output/result of the API.
ecode	NullOrInteger the error code, in case of no error NULL.
emessage	NullOrIntegerCharacter the error message, in case of no error NULL.
id	character identifier to API-call.

See Also

Other bitcoind functions: [CONRPC-class](#), [NullOrCharacter-class](#), [NullOrInteger-class](#), [conrpc\(\)](#), [rpcpost\(\)](#), [startbch\(\)](#), [startbtc\(\)](#), [stopbch\(\)](#), [stopbtc\(\)](#)

base58CheckDecode *Base 58 binary-to-text-decoding*

Description

This is a modified binary-to-text decoding used for decoding Bitcoin addresses, aka *Base58Check*. If this is applied to a WIF address and the first and last four bytes are dropped, the result is the corresponding private key.

Usage

```
base58CheckDecode(x)
```

Arguments

x character, string in hex format.

Value

list, the decoded elements of the string.

Author(s)

Bernhard Pfaff

References

https://en.bitcoin.it/wiki/Wallet_import_format,
<https://en.bitcoin.it/wiki/Address>,
https://en.bitcoin.it/wiki/Base58Check_encoding

See Also

Other BchAddresses: [BCHADR-class](#), [BTCADR-class](#), [PubHash2BchAdr\(\)](#), [PubKey2PubHash\(\)](#), [base58CheckEncode\(\)](#), [concatHex\(\)](#), [decodeHex\(\)](#), [hash160\(\)](#), [hash256\(\)](#), [validBchAdr\(\)](#), [validBtcAdr\(\)](#)

base58CheckEncode *Base 58 binary-to-text-encoding*

Description

This is a modified binary-to-text encoding used for encoding Bitcoin addresses, aka *Base58Check*. If this is applied to an extended private key with its trailing check sum, then the result is the *Wallet Import Format*, (WIF).

Usage

base58CheckEncode(x)

Arguments

x character, string in hex format.

Value

character, the encoded string.

Author(s)

Bernhard Pfaff

References

https://en.bitcoin.it/wiki/Wallet_import_format,
<https://en.bitcoin.it/wiki/Address>,
https://en.bitcoin.it/wiki/Base58Check_encoding

See Also

Other BchAdresses: [BCHADR-class](#), [BTCADR-class](#), [PubHash2BchAdr\(\)](#), [PubKey2PubHash\(\)](#), [base58CheckDecode\(\)](#), [concatHex\(\)](#), [decodeHex\(\)](#), [hash160\(\)](#), [hash256\(\)](#), [validBchAdr\(\)](#), [validBtcAdr\(\)](#)

BCHADR-class

S4 class BCHADR

Description

S4-class for BCH addresses

Slots

pubkey character, the 512-bit public key.
pubhash character, the hashed public key.
bchadr character, the BCH address.
mainnet logical, whether mainnet or testnet.

Author(s)

Bernhard Pfaff

References

<https://en.bitcoin.it/wiki/Address>

See Also

Other BchAdresses: [BTCADR-class](#), [PubHash2BchAdr\(\)](#), [PubKey2PubHash\(\)](#), [base58CheckDecode\(\)](#), [base58CheckEncode\(\)](#), [concatHex\(\)](#), [decodeHex\(\)](#), [hash160\(\)](#), [hash256\(\)](#), [validBchAdr\(\)](#), [validBtcAdr\(\)](#)

bkfee

Compute fee in a block

Description

This function returns the fee of the coinbase transaction. Hereby, the mining reward has been deducted. Initially, the mining reward was 50 BCH and is halved every 210,000 blocks.

Usage

```
bkfee(con, height)
```

Arguments

con	CONRPC, configuration object.
height	integer, the height of the block.

Value

numeric

Author(s)

Bernhard Pfaff

See Also

Other UtilityFuncs: [blockattime\(\)](#), [blockstats\(\)](#), [date2int\(\)](#), [int2date\(\)](#), [intMaxDay\(\)](#), [intMinDay\(\)](#), [intRangeDay\(\)](#), [intRangePeriod\(\)](#), [timeofblock\(\)](#), [txfee\(\)](#), [txids\(\)](#), [txinids\(\)](#), [txstats\(\)](#), [utxoage\(\)](#), [utxotype\(\)](#), [utxovalue\(\)](#)

blockatime *Block height at time*

Description

This function returns the block heights closest to a provided date/time (time zone is GMT).

Usage

```
blockatime(con, targetdate)
```

Arguments

con	CONRPC, configuration object.
targetdate	POSIXct, the date/time of closest block heights.

Value

data.frame: the heights, the times and the time differences (in minutes) to the provided date/time.

Author(s)

Bernhard Pfaff

See Also

Other UtilityFuncs: [bkfee\(\)](#), [blockstats\(\)](#), [date2int\(\)](#), [int2date\(\)](#), [intMaxDay\(\)](#), [intMinDay\(\)](#), [intRangeDay\(\)](#), [intRangePeriod\(\)](#), [timeofblock\(\)](#), [txfee\(\)](#), [txids\(\)](#), [txinids\(\)](#), [txstats\(\)](#), [utxoage\(\)](#), [utxotype\(\)](#), [utxovalue\(\)](#)

blockstats *Obtaining statistics of a block*

Description

This function returns key statistics of a block's content, such as the time, the count of transactions, and summary statistics of the UTXOs.

Usage

```
blockstats(con, height, excoinbase = TRUE)
```

Arguments

con	CONRPC, configuration object.
height	integer, the block's height.
excoinbase	logical, whether coinbase transaction should be excluded (default is TRUE).

Value

An object of class `data.frame`

Author(s)

Bernhard Pfaff

See Also

Other UtilityFuncs: [bkfee\(\)](#), [blockattime\(\)](#), [date2int\(\)](#), [int2date\(\)](#), [intMaxDay\(\)](#), [intMinDay\(\)](#), [intRangeDay\(\)](#), [intRangePeriod\(\)](#), [timeofblock\(\)](#), [txfee\(\)](#), [txids\(\)](#), [txinids\(\)](#), [txstats\(\)](#), [utxoage\(\)](#), [utxotype\(\)](#), [utxvalue\(\)](#)

BTCADR-class

S4 class BTCADR (BTC alias)

Description

S4-class for BTC addresses

Slots

`pubkey` character, the 512-bit public key.

`pubhash` character, the hashed public key.

`btcadr` character, the BTC address.

`mainnet` logical, whether mainnet or testnet.

Author(s)

Bernhard Pfaff

References

<https://en.bitcoin.it/wiki/Address>

See Also

Other BchAdresses: [BCHADR-class](#), [PubHash2BchAdr\(\)](#), [PubKey2PubHash\(\)](#), [base58CheckDecode\(\)](#), [base58CheckEncode\(\)](#), [concatHex\(\)](#), [decodeHex\(\)](#), [hash160\(\)](#), [hash256\(\)](#), [validBchAdr\(\)](#), [validBtcAdr\(\)](#)

clearbanned	<i>RPC-JSON API: clearbanned</i>
-------------	----------------------------------

Description

Clear all banned IPs.

Usage

```
clearbanned(con)
```

Arguments

con object of class CONRPC.

Value

A S4-object of class ANSRPC.

Author(s)

Bernhard Pfaff

References

<https://bitcoin.org/en/developer-reference#clearbanned>, <https://bitcoin.org/en/developer-reference#remote-procedure-calls-rpcs>

See Also

Other Network RPCs: [addnode\(\)](#), [disconnectnode\(\)](#), [getaddednodeinfo\(\)](#), [getconnectioncount\(\)](#), [getnettotals\(\)](#), [getnetworkinfo\(\)](#), [getpeerinfo\(\)](#), [listbanned\(\)](#), [ping\(\)](#)

concatHex	<i>Concatenate two hex strings</i>
-----------	------------------------------------

Description

This function concatenates two hex strings, provided without the 0x prefix, and returns a list object of the associated integers.

Usage

```
concatHex(hex1, hex2)
```

Arguments

hex1 character, a hex string.
hex2 character, a hex string.

Value

list

Author(s)

Bernhard Pfaff

References

https://en.bitcoin.it/wiki/Wallet_import_format,
<https://en.bitcoin.it/wiki/Address>

See Also

Other BchAdresses: [BCHADR-class](#), [BTCADR-class](#), [PubHash2BchAdr\(\)](#), [PubKey2PubHash\(\)](#), [base58CheckDecode\(\)](#), [base58CheckEncode\(\)](#), [decodeHex\(\)](#), [hash160\(\)](#), [hash256\(\)](#), [validBchAdr\(\)](#), [validBtcAdr\(\)](#)

conrpc

Extracting Configuration Settings

Description

This function extracts information from the configuration file `bitcoin.conf` with respect to the options `rpcuser` and `rpcpassword`.

Usage

```
conrpc(conf.file)
```

Arguments

conf.file character, the fully qualified path.

Value

An S4-object of class CONRPC.

Author(s)

Bernhard Pfaff

See Also

Other bitcoind functions: [ANSRPC-class](#), [CONRPC-class](#), [NullOrCharacter-class](#), [NullOrInteger-class](#), [rpcpost\(\)](#), [startbch\(\)](#), [startbtc\(\)](#), [stopbch\(\)](#), [stopbtc\(\)](#)

CONRPC-class

The CONRPC class

Description

S4-class for curl connections to RPC-JSON.

Details

The slots `rpcuse` and `rpcpwd` are required in the call to `curl`. Furthermore, the fully qualified path to `bitcoin.conf` (slot `config`) is required for starting and stopping bitcoind as daemon.

See Also

Other bitcoind functions: [ANSRPC-class](#), [NullOrCharacter-class](#), [NullOrInteger-class](#), [conrpc\(\)](#), [rpcpost\(\)](#), [startbch\(\)](#), [startbtc\(\)](#), [stopbch\(\)](#), [stopbtc\(\)](#)

containsPoint

containsPoint-methods

Description

Checks whether a point is on a defined elliptic curve.

Usage

```
containsPoint(curve, x, y)
```

```
## S4 method for signature 'ECPARAM,bigz,bigz'
containsPoint(curve, x, y)
```

```
## S4 method for signature 'ECPARAM,integer,integer'
containsPoint(curve, x, y)
```

```
## S4 method for signature 'ECPARAM,character,character'
containsPoint(curve, x, y)
```

Arguments

<code>curve</code>	an S4-object of class <code>ECPARAM</code> .
<code>x</code>	an S4-object of class <code>bigz</code> , the x-coordinate.
<code>y</code>	an S4-object of class <code>bigz</code> , the y-coordinate.

Value

logical

Author(s)

Bernhard Pfaff

References<https://en.bitcoin.it/wiki/Secp256k1>**See Also**Other EllipticCurve: [ECPARAM-class](#), [ECPOINT-class](#), [ECPARAMORNULL-class](#), [ecoperators](#), [ecparam\(\)](#), [ecpoint\(\)](#), [isNull\(\)](#)**Examples**

```
p <- "0xFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFC2F"
b <- "0x0000000000000000000000000000000000000000000000000000000000000007"
a <- "0x00000000000000000000000000000000000000000000000000000000000000"
curve256 <- ecparam(p, a, b)
Gx <- "0x79BE667EF9DCBBAC55A06295CE870B07029BFCD82DCE28D959F2815B16F81798"
Gy <- "0x483ada7726a3c4655da4fbfc0e1108a8fd17b448a68554199c47d08ffb10d4b8"
containsPoint(curve256, Gx, Gy)
```

date2int

*Convert date/time to integer***Description**

This function returns the associated integer time for a given date/time object (coercible as POSIXct object).

Usage

date2int(x)

Arguments

x POSIXct, date/time object.

Value

integer

Author(s)

Bernhard Pfaff

See Also

Other UtilityFuncs: [bkfee\(\)](#), [blockattime\(\)](#), [blockstats\(\)](#), [int2date\(\)](#), [intMaxDay\(\)](#), [intMinDay\(\)](#), [intRangeDay\(\)](#), [intRangePeriod\(\)](#), [timeofblock\(\)](#), [txfee\(\)](#), [txids\(\)](#), [txinids\(\)](#), [txstats\(\)](#), [utxoage\(\)](#), [utxotype\(\)](#), [utxovalue\(\)](#)

Examples

```
d <- "2017-03-15"  
date2int(d)
```

decodeHex

Decoding of a hex string

Description

This function converts a hex string,, whereby the string must not contain the 0x prefix, to a list object with the associated integers as its elements.

Usage

```
decodeHex(s)
```

Arguments

s character, the hex string.

Value

list

Author(s)

Bernhard Pfaff

References

https://en.bitcoin.it/wiki/Wallet_import_format,
<https://en.bitcoin.it/wiki/Address>

See Also

Other BchAdresses: [BCHADR-class](#), [BTCADR-class](#), [PubHash2BchAdr\(\)](#), [PubKey2PubHash\(\)](#), [base58CheckDecode\(\)](#), [base58CheckEncode\(\)](#), [concatHex\(\)](#), [hash160\(\)](#), [hash256\(\)](#), [validBchAdr\(\)](#), [validBtcAdr\(\)](#)

decoderawtransaction *RPC-JSON API: decoderawtransaction*

Description

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

Usage

```
decoderawtransaction(con, hexstring)
```

Arguments

con	object of class CONRPC.
hexstring	character, the transaction hex string.

Value

A S4-object of class ANSRPC.

Author(s)

Bernhard Pfaff

References

<https://bitcoin.org/en/developer-reference#getblock>, <https://bitcoin.org/en/developer-reference#remote-procedure-calls-rpcs>

See Also

Other RawTransactions RPCs: [getrawtransaction\(\)](#)

decodescript *RPC-JSON API: decodescript*

Description

The decodescript RPC decodes a hex-encoded P2SH redeem script.

Usage

```
decodescript(con, redeem)
```

Arguments

con	object of class CONRPC.
redeem	character, the P2SH.

Value

A S4-object of class ANSRPC.

Author(s)

Bernhard Pfaff

References

<https://bitcoin.org/en/developer-reference#decodescript>, <https://bitcoin.org/en/developer-reference#remote-procedure-calls-rpcs>

See Also

Other Blockchain RPCs: [getbestblockhash\(\)](#), [getblockchaininfo\(\)](#), [getblockcount\(\)](#), [getblockhash\(\)](#), [getblockheader\(\)](#), [getblock\(\)](#), [getchaintips\(\)](#), [getchaintxstats\(\)](#), [getdifficulty\(\)](#), [getmempoolancestors\(\)](#), [getmempooldescendants\(\)](#), [getmempoolentry\(\)](#), [getmempoolinfo\(\)](#), [getrawmempool\(\)](#), [gettxoutproof\(\)](#), [gettxoutsetinfo\(\)](#), [gettxout\(\)](#), [pruneblockchain\(\)](#), [verifychain\(\)](#), [verifytxoutproof\(\)](#)

disconnectnode	<i>RPC-JSON API: disconnectnode</i>
----------------	-------------------------------------

Description

Immediately disconnects from the specified peer node. Strictly one out of address and nodeid can be provided to identify the node.

Usage

```
disconnectnode(con, address = NULL, nodeid = NULL)
```

Arguments

con	object of class CONRPC.
address	character the IP address/port of the node.
nodeid	character The node ID (see getpeerinfo() for node IDs).

Value

A S4-object of class ANSRPC.

Author(s)

Bernhard Pfaff

References

<https://bitcoin.org/en/developer-reference#disconnectnode>, <https://bitcoin.org/en/developer-reference#remote-procedure-calls-rpcs>

See Also

Other Network RPCs: [addnode\(\)](#), [clearbanned\(\)](#), [getaddednodeinfo\(\)](#), [getconnectioncount\(\)](#), [getnettotals\(\)](#), [getnetworkinfo\(\)](#), [getpeerinfo\(\)](#), [listbanned\(\)](#), [ping\(\)](#)

ecoperators

Elliptic curve operators

Description

The following operations for EC points are available:

- doubleUpmultiplying a point by itself
- +point addition
- leftmostBithighest bit value of an integer
- ANDlogical and-operator for two integers
- *multiplication of an integer scalar with an EC point

Usage

```
doubleUp(ecp)

## S4 method for signature 'ECPOINT'
doubleUp(ecp)

## S4 method for signature 'ECPOINT,ECPOINT'
e1 + e2

leftmostBit(x)

## S4 method for signature 'bigz'
leftmostBit(x)

AND(x, y)

## S4 method for signature 'bigz,bigz'
AND(x, y)
```

```
## S4 method for signature 'ECPOINT,bigz'  
e1 * e2
```

```
## S4 method for signature 'bigz,ECPOINT'  
e1 * e2
```

Arguments

eCP	point on elliptic curve
e1	point on elliptic curve, or integer
e2	point on elliptic curve, or integer
x	integer
y	integer

Author(s)

Bernhard Pfaff

References

<https://en.bitcoin.it/wiki/Secp256k1>

See Also

Other EllipticCurve: [ECPARAM-class](#), [ECPOINT-class](#), [ECPARAMorNULL-class](#), [containsPoint\(\)](#), [ecparam\(\)](#), [ecpoint\(\)](#), [isNull\(\)](#)

ecparam

Creating objects of class ECPARAM

Description

This function returns an object of S4-class ECPARAM, that does contain the parametrization of an elliptic curve.

Usage

```
ecparam(p, a, b)
```

Arguments

p	integer
a	integer
b	integer

Value

An object of S4-class ECPARAM

Author(s)

Bernhard Pfaff

References

<https://en.bitcoin.it/wiki/Secp256k1>

See Also

Other EllipticCurve: [ECPARAM-class](#), [ECPOINT-class](#), [EcparamOrNull-class](#), [containsPoint\(\)](#), [ecoperators](#), [ecpoint\(\)](#), [isNull\(\)](#)

Examples

```
p <- "0xFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFC2F"
b <- "0x0000000000000000000000000000000000000000000000000000000000000007"
a <- "0x00000000000000000000000000000000000000000000000000000000000000"
curve256 <- ecparam(p, a, b)
curve256
```

ECPARAM-class

The ECPARAM class

Description

S4-class for elliptic curve parameters. Objects of this class do contain the big integer parameters of elliptic curves. Instances of this class are ordinarily created by a call to `ecparam`

Slots

p bigz, curve dimension.

a bigz, parameter.

b bigz, parameter.

Author(s)

Bernhard Pfaff

References

<https://en.bitcoin.it/wiki/Secp256k1>

See Also

Other EllipticCurve: [ECPOINT-class](#), [EcpaamOrNull-class](#), [containsPoint\(\)](#), [ecoperators](#), [ecparam\(\)](#), [ecpoint\(\)](#), [isNull\(\)](#)

`EcpaamOrNull-class` *S4 Class Union ECPARAM or NULL*

Description

S4-class union of NULL or ECPARAM.

Author(s)

Bernhard Pfaff

References

<https://en.bitcoin.it/wiki/Secp256k1>

See Also

Other EllipticCurve: [ECPARAM-class](#), [ECPOINT-class](#), [containsPoint\(\)](#), [ecoperators](#), [ecparam\(\)](#), [ecpoint\(\)](#), [isNull\(\)](#)

`ecpoint` *Creating objects of class ECPOINT*

Description

This function returns an object of S4-class ECPOINT, that does represent a point on an elliptic curve.

Usage

```
ecpoint(ecparam = NULL, x, y, r = NULL)
```

Arguments

<code>ecparam</code>	integerECPARAM
<code>x</code>	x-coordinate, to be coercible to bigz.
<code>y</code>	y-coordinate, to be coercible to bigz.
<code>r</code>	the order of the base point.

Value

An object of S4-class ECPOINT

Author(s)

Bernhard Pfaff

References<https://en.bitcoin.it/wiki/Secp256k1>**See Also**Other EllipticCurve: [ECPARAM-class](#), [ECPOINT-class](#), [EccparamOrNull-class](#), [containsPoint\(\)](#), [ecoperators](#), [ecparam\(\)](#), [isNull\(\)](#)**Examples**

```

p <- "0xFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFC2F"
b <- "0x0000000000000000000000000000000000000000000000000000000000000007"
a <- "0x00000000000000000000000000000000000000000000000000000000000000"
r <- "0xFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFEBAAEDCE6AF48A03BBFD25E8CD0364141"
x <- "0x79BE667EF9DCBBAC55A06295CE870B07029BFCDB2DCE28D959F2815B16F81798"
y <- "0x483ada7726a3c4655da4fbfc0e1108a8fd17b448a68554199c47d08ffb10d4b8"
curve256 <- ecparam(p, a, b)
ecp <- ecpoint(curve256, x, y, r)
ecp

```

ECPOINT-class

S4 Class ECPOINT

Description

S4-class for a point on an elliptic curve. Ordinarily, objects are created by calling `ecpoint`.

Slots

```

ecparam ECPARAM
x bigz
y bigz
r bigz

```

Author(s)

Bernhard Pfaff

References<https://en.bitcoin.it/wiki/Secp256k1>

See Also

Other EllipticCurve: [ECPARAM-class](#), [EcparamOrNull-class](#), [containsPoint\(\)](#), [ecoperators](#), [ecparam\(\)](#), [ecpoint\(\)](#), [isNull\(\)](#)

getaddednodeinfo *RPC-JSON API: getaddednodeinfo*

Description

Returns information about the given added node, or all added nodes (note that onetry addnodes are not listed here)

Usage

```
getaddednodeinfo(con, node = NULL, dns = FALSE)
```

Arguments

con	object of class CONRPC.
node	character the node (see getpeerinfo() for nodes).
dns	logical If FALSE, only a list of added nodes will be provided, otherwise connected information will also be available.

Value

A S4-object of class ANSRPC.

Author(s)

Bernhard Pfaff

References

<https://bitcoin.org/en/developer-reference#getaddednodeinfo>, <https://bitcoin.org/en/developer-reference#remote-procedure-calls-rpcs>

See Also

Other Network RPCs: [addnode\(\)](#), [clearbanned\(\)](#), [disconnectnode\(\)](#), [getconnectioncount\(\)](#), [getnettotals\(\)](#), [getnetworkinfo\(\)](#), [getpeerinfo\(\)](#), [listbanned\(\)](#), [ping\(\)](#)

getbestblockhash *RPC-JSON API: getbestblockhash*

Description

Returns the hash of the best (tip) block in the longest blockchain.

Usage

```
getbestblockhash(con)
```

Arguments

con object of class CONRPC.

Value

A S4-object of class ANSRPC.

Author(s)

Bernhard Pfaff

References

<https://bitcoin.org/en/developer-reference#getbestblockhash>, <https://bitcoin.org/en/developer-reference#remote-procedure-calls-rpcs>

See Also

Other Blockchain RPCs: [decodescript\(\)](#), [getblockchaininfo\(\)](#), [getblockcount\(\)](#), [getblockhash\(\)](#), [getblockheader\(\)](#), [getblock\(\)](#), [getchaintips\(\)](#), [getchaintxstats\(\)](#), [getdifficulty\(\)](#), [getmempoolancestors\(\)](#), [getmempooldescendants\(\)](#), [getmempoolentry\(\)](#), [getmempoolinfo\(\)](#), [getrawmempool\(\)](#), [gettxoutproof\(\)](#), [gettxoutsetinfo\(\)](#), [gettxout\(\)](#), [pruneblockchain\(\)](#), [verifychain\(\)](#), [verifytxoutproof\(\)](#)

getblock *RPC-JSON API: getblock*

Description

Returns information of a block hash. The returned level of details depends on the argument verbosity.

Usage

```
getblock(con, blockhash, verbosity = c("11", "10", "12"))
```

Arguments

con	object of class CONRPC.
blockhash	character, the block hash.
verbosity	character, level of returned details.

Value

A S4-object of class ANSRPC.

Details If verbosity is 'l0', returns a string that is serialized, hex-encoded data for block 'hash'. If verbosity is 'l1' (the default), returns an object with information about block <hash>. If verbosity is 'l2', returns an object with information about block <hash> and information about each transaction.

Author(s)

Bernhard Pfaff

References

<https://bitcoin.org/en/developer-reference#getblock>, <https://bitcoin.org/en/developer-reference#remote-procedure-calls-rpcs>

See Also

Other Blockchain RPCs: [decodescript\(\)](#), [getbestblockhash\(\)](#), [getblockchaininfo\(\)](#), [getblockcount\(\)](#), [getblockhash\(\)](#), [getblockheader\(\)](#), [getchaintips\(\)](#), [getchaintxstats\(\)](#), [getdifficulty\(\)](#), [getmempoolancestors\(\)](#), [getmempooldescendants\(\)](#), [getmempoolentry\(\)](#), [getmempoolinfo\(\)](#), [getrawmempool\(\)](#), [gettxoutproof\(\)](#), [gettxoutsetinfo\(\)](#), [gettxout\(\)](#), [pruneblockchain\(\)](#), [verifychain\(\)](#), [verifytxoutproof\(\)](#)

getblockchaininfo *RPC-JSON API: getblockchaininfo*

Description

Returns an object containing various state info regarding blockchain processing.

Usage

```
getblockchaininfo(con)
```

Arguments

con	object of class CONRPC.
-----	-------------------------

Value

A S4-object of class ANSRPC.

Author(s)

Bernhard Pfaff

References

<https://bitcoin.org/en/developer-reference#getblockchaininfo>, <https://bitcoin.org/en/developer-reference#remote-procedure-calls-rpcs>

See Also

Other Blockchain RPCs: [decodescript\(\)](#), [getbestblockhash\(\)](#), [getblockcount\(\)](#), [getblockhash\(\)](#), [getblockheader\(\)](#), [getblock\(\)](#), [getchaintips\(\)](#), [getchaintxstats\(\)](#), [getdifficulty\(\)](#), [getmempoolancestors\(\)](#), [getmempooldescendants\(\)](#), [getmempoolentry\(\)](#), [getmempoolinfo\(\)](#), [getrawmempool\(\)](#), [gettxoutproof\(\)](#), [gettxoutsetinfo\(\)](#), [gettxout\(\)](#), [pruneblockchain\(\)](#), [verifychain\(\)](#), [verifytxoutproof\(\)](#)

getblockcount

RPC-JSON API: getblockcount

Description

Returns the number of blocks in the longest blockchain.

Usage

```
getblockcount(con)
```

Arguments

con object of class CONRPC.

Value

A S4-object of class ANSRPC.

Author(s)

Bernhard Pfaff

References

<https://bitcoin.org/en/developer-reference#getblockcount>, <https://bitcoin.org/en/developer-reference#remote-procedure-calls-rpcs>

See Also

Other Blockchain RPCs: [decodescript\(\)](#), [getbestblockhash\(\)](#), [getblockchaininfo\(\)](#), [getblockhash\(\)](#), [getblockheader\(\)](#), [getblock\(\)](#), [getchaintips\(\)](#), [getchaintxstats\(\)](#), [getdifficulty\(\)](#), [getmempoolancestors\(\)](#), [getmempooldescendants\(\)](#), [getmempoolentry\(\)](#), [getmempoolinfo\(\)](#), [getrawmempool\(\)](#), [gettxoutproof\(\)](#), [gettxoutsetinfo\(\)](#), [gettxout\(\)](#), [pruneblockchain\(\)](#), [verifychain\(\)](#), [verifytxoutproof\(\)](#)

getblockhash	<i>RPC-JSON API: getblockhash</i>
--------------	-----------------------------------

Description

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

Usage

```
getblockhash(con, height)
```

Arguments

con	object of class CONRPC.
height	integer the height index.

Value

A S4-object of class ANSRPC.

Author(s)

Bernhard Pfaff

References

<https://bitcoin.org/en/developer-reference#getblockhash>, <https://bitcoin.org/en/developer-reference#remote-procedure-calls-rpcs>

See Also

Other Blockchain RPCs: [decodescript\(\)](#), [getbestblockhash\(\)](#), [getblockchaininfo\(\)](#), [getblockcount\(\)](#), [getblockheader\(\)](#), [getblock\(\)](#), [getchaintips\(\)](#), [getchaintxstats\(\)](#), [getdifficulty\(\)](#), [getmempoolancestors\(\)](#), [getmempooldescendants\(\)](#), [getmempoolentry\(\)](#), [getmempoolinfo\(\)](#), [getrawmempool\(\)](#), [gettxoutproof\(\)](#), [gettxoutsetinfo\(\)](#), [gettxout\(\)](#), [pruneblockchain\(\)](#), [verifychain\(\)](#), [verifytxoutproof\(\)](#)

getblockheader	<i>RPC-JSON API: getblockheader</i>
----------------	-------------------------------------

Description

Returns the block header for a given hash string.

Usage

```
getblockheader(con, hash, verbose = TRUE)
```

Arguments

con	object of class CONRPC.
hash	character the block hash.
verbose	logical TRUE for a json object, FALSE for the hex encoded data.

Value

A S4-object of class ANSRPC.

Details

If verbose is false, returns a string that is serialized, hex-encoded data for blockheader 'hash'. If verbose is true, returns an Object with information about blockheader <hash>.

Author(s)

Bernhard Pfaff

References

<https://bitcoin.org/en/developer-reference#getblockheader>, <https://bitcoin.org/en/developer-reference#remote-procedure-calls-rpcs>

See Also

Other Blockchain RPCs: [decodescript\(\)](#), [getbestblockhash\(\)](#), [getblockchaininfo\(\)](#), [getblockcount\(\)](#), [getblockhash\(\)](#), [getblock\(\)](#), [getchaintips\(\)](#), [getchaintxstats\(\)](#), [getdifficulty\(\)](#), [getmempoolancestors\(\)](#), [getmempooldescendants\(\)](#), [getmempoolentry\(\)](#), [getmempoolinfo\(\)](#), [getrawmempool\(\)](#), [gettxoutproof\(\)](#), [gettxoutsetinfo\(\)](#), [gettxout\(\)](#), [pruneblockchain\(\)](#), [verifychain\(\)](#), [verifytxoutproof\(\)](#)

getchaintips	<i>RPC-JSON API: getchaintips</i>
--------------	-----------------------------------

Description

Return information about all known tips in the block tree, including the main chain as well as orphaned branches.

Usage

```
getchaintips(con)
```

Arguments

con object of class CONRPC.

Value

A S4-object of class ANSRPC.

Author(s)

Bernhard Pfaff

References

<https://bitcoin.org/en/developer-reference#getchaintips>, <https://bitcoin.org/en/developer-reference#remote-procedure-calls-rpcs>

See Also

Other Blockchain RPCs: [decodescript\(\)](#), [getbestblockhash\(\)](#), [getblockchaininfo\(\)](#), [getblockcount\(\)](#), [getblockhash\(\)](#), [getblockheader\(\)](#), [getblock\(\)](#), [getchaintxstats\(\)](#), [getdifficulty\(\)](#), [getmempoolancestors\(\)](#), [getmempooldescendants\(\)](#), [getmempoolentry\(\)](#), [getmempoolinfo\(\)](#), [getrawmempool\(\)](#), [gettxoutproof\(\)](#), [gettxoutsetinfo\(\)](#), [gettxout\(\)](#), [pruneblockchain\(\)](#), [verifychain\(\)](#), [verifytxoutproof\(\)](#)

getchaintxstats	<i>RPC-JSON API: getchaintxstats</i>
-----------------	--------------------------------------

Description

Compute statistics about the total number and rate of transactions in the chain.

Usage

```
getchaintxstats(con, nblocks = NULL, blockhash = NULL)
```

Arguments

- con object of class CONRPC.
- nblocks integer optional, size of the window in number of blocks (default: one month).
- blockhash character optional, the hash of the block that ends the window.

Value

A S4-object of class ANSRPC.

Author(s)

Bernhard Pfaff

References

<https://bitcoin.org/en/developer-reference#getchaintxstats>, <https://bitcoin.org/en/developer-reference#remote-procedure-calls-rpcs>

See Also

Other Blockchain RPCs: [decodescript\(\)](#), [getbestblockhash\(\)](#), [getblockchaininfo\(\)](#), [getblockcount\(\)](#), [getblockhash\(\)](#), [getblockheader\(\)](#), [getblock\(\)](#), [getchaintips\(\)](#), [getdifficulty\(\)](#), [getmempoolancestors\(\)](#), [getmempooldescendants\(\)](#), [getmempoolentry\(\)](#), [getmempoolinfo\(\)](#), [getrawmempool\(\)](#), [gettxoutproof\(\)](#), [gettxoutsetinfo\(\)](#), [gettxout\(\)](#), [pruneblockchain\(\)](#), [verifychain\(\)](#), [verifytxoutproof\(\)](#)

getconnectioncount *RPC-JSON API: getconnectioncount*

Description

Returns the number of connections to other nodes.

Usage

`getconnectioncount(con)`

Arguments

- con object of class CONRPC.

Value

A S4-object of class ANSRPC.

Author(s)

Bernhard Pfaff

References

<https://bitcoin.org/en/developer-reference#getconnectioncount>, <https://bitcoin.org/en/developer-reference#remote-procedure-calls-rpcs>

See Also

Other Network RPCs: [addnode\(\)](#), [clearbanned\(\)](#), [disconnectnode\(\)](#), [getaddednodeinfo\(\)](#), [getnettotals\(\)](#), [getnetworkinfo\(\)](#), [getpeerinfo\(\)](#), [listbanned\(\)](#), [ping\(\)](#)

getdifficulty

RPC-JSON API: getdifficulty

Description

Returns the proof-of-work difficulty as a multiple of the minimum difficulty.

Usage

```
getdifficulty(con)
```

Arguments

con object of class CONRPC.

Value

A S4-object of class ANSRPC.

Author(s)

Bernhard Pfaff

References

<https://bitcoin.org/en/developer-reference#getdifficulty>, <https://bitcoin.org/en/developer-reference#remote-procedure-calls-rpcs>

See Also

Other Blockchain RPCs: [decodescript\(\)](#), [getbestblockhash\(\)](#), [getblockchaininfo\(\)](#), [getblockcount\(\)](#), [getblockhash\(\)](#), [getblockheader\(\)](#), [getblock\(\)](#), [getchaintips\(\)](#), [getchaintxstats\(\)](#), [getmempoolancestors\(\)](#), [getmempooldescendants\(\)](#), [getmempoolentry\(\)](#), [getmempoolinfo\(\)](#), [getrawmempool\(\)](#), [gettxoutproof\(\)](#), [gettxoutsetinfo\(\)](#), [gettxout\(\)](#), [pruneblockchain\(\)](#), [verifychain\(\)](#), [verifytxoutproof\(\)](#)

gethelp *RPC-JSON API: Help*

Description

Returning information about RPC functions.

Usage

```
gethelp(con, rpc = "")
```

Arguments

con	object of class CONRPC.
rpc	character, name of RPC function.

Value

A S4-object of class ANSRPC.

Author(s)

Bernhard Pfaff

References

<https://bitcoin.org/en/developer-reference#help>, <https://bitcoin.org/en/developer-reference#remote-procedure-calls-rpcs>

See Also

Other Control RPCs: [getinfo\(\)](#), [getwalletinfo\(\)](#)

getinfo *RPC-JSON API: getinfo*

Description

Returning information about bitcoin configuration and settings.

Usage

```
getinfo(con)
```

Arguments

con	object of class CONRPC.
-----	-------------------------

Details

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.

Value

A S4-object of class ANSRPC.

Author(s)

Bernhard Pfaff

References

<https://bitcoin.org/en/developer-reference#getinfo>, <https://bitcoin.org/en/developer-reference#remote-procedure-calls-rpcs>

See Also

Other Control RPCs: [gethelp\(\)](#), [getwalletinfo\(\)](#)

getmempoolancestors *RPC-JSON API: getmempoolancestors*

Description

If txid is in the mempool, returns all in-mempool ancestors.

Usage

```
getmempoolancestors(con, txid, verbose = FALSE)
```

Arguments

con	object of class CONRPC.
txid	character, the transaction id (must be in mempool).
verbose	logical, TrueTRUE for a json object, FALSE for array of transaction ids (default).

Value

A S4-object of class ANSRPC.

Author(s)

Bernhard Pfaff

References

<https://bitcoin.org/en/developer-reference#getmempoolancestors>, <https://bitcoin.org/en/developer-reference#remote-procedure-calls-rpcs>

See Also

Other Blockchain RPCs: [decodescript\(\)](#), [getbestblockhash\(\)](#), [getblockchaininfo\(\)](#), [getblockcount\(\)](#), [getblockhash\(\)](#), [getblockheader\(\)](#), [getblock\(\)](#), [getchaintips\(\)](#), [getchaintxstats\(\)](#), [getdifficulty\(\)](#), [getmempooldescendants\(\)](#), [getmempoolentry\(\)](#), [getmempoolinfo\(\)](#), [getrawmempool\(\)](#), [gettxoutproof\(\)](#), [gettxoutsetinfo\(\)](#), [gettxout\(\)](#), [pruneblockchain\(\)](#), [verifychain\(\)](#), [verifytxoutproof\(\)](#)

getmempooldescendants *RPC-JSON API: getmempooldescendants*

Description

If txid is in the mempool, returns all in-mempool descendants.

Usage

```
getmempooldescendants(con, txid, verbose = FALSE)
```

Arguments

con	object of class CONRPC.
txid	character, the transaction id (must be in mempool).
verbose	logical, TrueTRUE for a json object, FALSE for array of transaction ids (default).

Value

A S4-object of class ANSRPC.

Author(s)

Bernhard Pfaff

References

<https://bitcoin.org/en/developer-reference#getmempooldescendants>, <https://bitcoin.org/en/developer-reference#remote-procedure-calls-rpcs>

See Also

Other Blockchain RPCs: [decodescript\(\)](#), [getbestblockhash\(\)](#), [getblockchaininfo\(\)](#), [getblockcount\(\)](#), [getblockhash\(\)](#), [getblockheader\(\)](#), [getblock\(\)](#), [getchaintips\(\)](#), [getchaintxstats\(\)](#), [getdifficulty\(\)](#), [getmempoolancestors\(\)](#), [getmempoolentry\(\)](#), [getmempoolinfo\(\)](#), [getrawmempool\(\)](#), [gettxoutproof\(\)](#), [gettxoutsetinfo\(\)](#), [gettxout\(\)](#), [pruneblockchain\(\)](#), [verifychain\(\)](#), [verifytxoutproof\(\)](#)

getmempoolentry	<i>RPC-JSON API: getmempoolentry</i>
-----------------	--------------------------------------

Description

Returns mempool data for given transaction.

Usage

```
getmempoolentry(con, txid)
```

Arguments

con	object of class CONRPC.
txid	character, the transaction id (must be in mempool).

Value

A S4-object of class ANSRPC.

Author(s)

Bernhard Pfaff

References

<https://bitcoin.org/en/developer-reference#getmempoolentry>, <https://bitcoin.org/en/developer-reference#remote-procedure-calls-rpcs>

See Also

Other Blockchain RPCs: [decodescript\(\)](#), [getbestblockhash\(\)](#), [getblockchaininfo\(\)](#), [getblockcount\(\)](#), [getblockhash\(\)](#), [getblockheader\(\)](#), [getblock\(\)](#), [getchaintips\(\)](#), [getchaintxstats\(\)](#), [getdifficulty\(\)](#), [getmempoolancestors\(\)](#), [getmempooldescendants\(\)](#), [getmempoolinfo\(\)](#), [getrawmempool\(\)](#), [gettxoutproof\(\)](#), [gettxoutsetinfo\(\)](#), [gettxout\(\)](#), [pruneblockchain\(\)](#), [verifychain\(\)](#), [verifytxoutproof\(\)](#)

getmempoolinfo	<i>RPC-JSON API: getmempoolinfo</i>
----------------	-------------------------------------

Description

Returns details on the active state of the TX memory pool.

Usage

```
getmempoolinfo(con)
```

Arguments

con object of class CONRPC.

Value

A S4-object of class ANSRPC.

Author(s)

Bernhard Pfaff

References

<https://bitcoin.org/en/developer-reference#getmempoolinfo>, <https://bitcoin.org/en/developer-reference#remote-procedure-calls-rpcs>

See Also

Other Blockchain RPCs: [decodescript\(\)](#), [getbestblockhash\(\)](#), [getblockchaininfo\(\)](#), [getblockcount\(\)](#), [getblockhash\(\)](#), [getblockheader\(\)](#), [getblock\(\)](#), [getchaintips\(\)](#), [getchaintxstats\(\)](#), [getdifficulty\(\)](#), [getmempoolancestors\(\)](#), [getmempooldescendants\(\)](#), [getmempoolentry\(\)](#), [getrawmempool\(\)](#), [gettxoutproof\(\)](#), [gettxoutsetinfo\(\)](#), [gettxout\(\)](#), [pruneblockchain\(\)](#), [verifychain\(\)](#), [verifytxoutproof\(\)](#)

getnettotals	<i>RPC-JSON API: getnettotals</i>
--------------	-----------------------------------

Description

Returns information about network traffic, including bytes in, bytes out, and current time.

Usage

```
getnettotals(con)
```

Arguments

con object of class CONRPC.

Value

A S4-object of class ANSRPC.

Author(s)

Bernhard Pfaff

References

<https://bitcoin.org/en/developer-reference#getnettotals>, <https://bitcoin.org/en/developer-reference#remote-procedure-calls-rpcs>

See Also

Other Network RPCs: [addnode\(\)](#), [clearbanned\(\)](#), [disconnectnode\(\)](#), [getaddednodeinfo\(\)](#), [getconnectioncount\(\)](#), [getnetworkinfo\(\)](#), [getpeerinfo\(\)](#), [listbanned\(\)](#), [ping\(\)](#)

getnetworkinfo

RPC-JSON API: getnetworkinfo

Description

Returns an object containing various state info regarding P2P networking.

Usage

```
getnetworkinfo(con)
```

Arguments

con object of class CONRPC.

Value

A S4-object of class ANSRPC.

Author(s)

Bernhard Pfaff

References

<https://bitcoin.org/en/developer-reference#getnetworkinfo>, <https://bitcoin.org/en/developer-reference#remote-procedure-calls-rpcs>

See Also

Other Network RPCs: [addnode\(\)](#), [clearbanned\(\)](#), [disconnectnode\(\)](#), [getaddednodeinfo\(\)](#), [getconnectioncount\(\)](#), [getnettotals\(\)](#), [getpeerinfo\(\)](#), [listbanned\(\)](#), [ping\(\)](#)

getpeerinfo

RPC-JSON API: getpeerinfo

Description

Returns data about each connected network node as a json array of objects.

Usage

```
getpeerinfo(con)
```

Arguments

con object of class CONRPC.

Value

A S4-object of class ANSRPC.

Author(s)

Bernhard Pfaff

References

<https://bitcoin.org/en/developer-reference#getpeerinfo>, <https://bitcoin.org/en/developer-reference#remote-procedure-calls-rpcs>

See Also

Other Network RPCs: [addnode\(\)](#), [clearbanned\(\)](#), [disconnectnode\(\)](#), [getaddednodeinfo\(\)](#), [getconnectioncount\(\)](#), [getnettotals\(\)](#), [getnetworkinfo\(\)](#), [listbanned\(\)](#), [ping\(\)](#)

getrawmempool	<i>RPC-JSON API: getrawmempool</i>
---------------	------------------------------------

Description

Returns all transaction ids in memory pool as a json array of string transaction ids. Hint: use `getmempoolentry` to fetch a specific transaction from the mempool.

Usage

```
getrawmempool(con, verbose = TRUE)
```

Arguments

<code>con</code>	object of class <code>CONRPC</code> .
<code>verbose</code>	logical, TRUE for a json object, FALSE for array of transaction ids

Value

A S4-object of class `ANSRPC`.

Author(s)

Bernhard Pfaff

References

<https://bitcoin.org/en/developer-reference#getrawmempool>, <https://bitcoin.org/en/developer-reference#remote-procedure-calls-rpcs>

See Also

Other Blockchain RPCs: `decodescript()`, `getbestblockhash()`, `getblockchaininfo()`, `getblockcount()`, `getblockhash()`, `getblockheader()`, `getblock()`, `getchaintips()`, `getchaintxstats()`, `getdifficulty()`, `getmempoolancestors()`, `getmempooldescendants()`, `getmempoolentry()`, `getmempoolinfo()`, `gettxoutproof()`, `gettxoutsetinfo()`, `gettxout()`, `pruneblockchain()`, `verifychain()`, `verifytxoutproof()`

getrawtransaction *RPC-JSON API: getrawtransaction*

Description

Returns the raw transaction data.

Usage

```
getrawtransaction(con, txid, verbose = FALSE)
```

Arguments

con	object of class CONRPC.
txid	character, the transaction id.
verbose	logical, type of output.

Value

A S4-object of class ANSRPC.

Details By default this function only works for mempool transactions. If the `-txindex` option is enabled, it also works for blockchain transactions. **DEPRECATED:** for now, it also works for transactions with unspent outputs. If `verbose` is `'true'`, returns an object with information about `'txid'`. If `verbose` is `'false'` or omitted, returns a string that is serialized, hex-encoded data for `'txid'`.

Author(s)

Bernhard Pfaff

References

<https://bitcoin.org/en/developer-reference#getblock>, <https://bitcoin.org/en/developer-reference#remote-procedure-calls-rpcs>

See Also

Other RawTransactions RPCs: [decoderawtransaction\(\)](#)

`gettxout`*RPC-JSON API: gettxout*

Description

Returns details about an unspent transaction output.

Usage

```
gettxout(con, txid, n, incmempool = TRUE)
```

Arguments

<code>con</code>	object of class <code>CONRPC</code> .
<code>txid</code>	character the transaction id.
<code>n</code>	integer vout number.
<code>incmempool</code>	logical whether to include the mempool (default <code>TRUE</code>).

Details

Note that an unspent output that is spent in the mempool won't appear.

Value

A S4-object of class `ANSRPC`.

Author(s)

Bernhard Pfaff

References

<https://bitcoin.org/en/developer-reference#gettxout>, <https://bitcoin.org/en/developer-reference#remote-procedure-calls-rpcs>

See Also

Other Blockchain RPCs: [decodescript\(\)](#), [getbestblockhash\(\)](#), [getblockchaininfo\(\)](#), [getblockcount\(\)](#), [getblockhash\(\)](#), [getblockheader\(\)](#), [getblock\(\)](#), [getchaintips\(\)](#), [getchaintxstats\(\)](#), [getdifficulty\(\)](#), [getmempoolancestors\(\)](#), [getmempooldescendants\(\)](#), [getmempoolentry\(\)](#), [getmempoolinfo\(\)](#), [getrawmempool\(\)](#), [gettxoutproof\(\)](#), [gettxoutsetinfo\(\)](#), [pruneblockchain\(\)](#), [verifychain\(\)](#), [verifytxoutproof\(\)](#)

gettxoutproof	<i>RPC-JSON API: gettxoutproof</i>
---------------	------------------------------------

Description

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

Usage

```
gettxoutproof(con, txids, blockhash = NULL)
```

Arguments

con	object of class CONRPC.
txids	character a json array of txids to filter.
blockhash	integer looks for txid in the block with this hash, (optional, default NULL).

Details

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).

Value

A S4-object of class ANSRPC.

Author(s)

Bernhard Pfaff

References

<https://bitcoin.org/en/developer-reference#gettxoutproof>, <https://bitcoin.org/en/developer-reference#remote-procedure-calls-rpcs>

See Also

Other Blockchain RPCs: [decodescript\(\)](#), [getbestblockhash\(\)](#), [getblockchaininfo\(\)](#), [getblockcount\(\)](#), [getblockhash\(\)](#), [getblockheader\(\)](#), [getblock\(\)](#), [getchaintips\(\)](#), [getchaintxstats\(\)](#), [getdifficulty\(\)](#), [getmempoolancestors\(\)](#), [getmempooldescendants\(\)](#), [getmempoolentry\(\)](#), [getmempoolinfo\(\)](#), [getrawmempool\(\)](#), [gettxoutsetinfo\(\)](#), [gettxout\(\)](#), [pruneblockchain\(\)](#), [verifychain\(\)](#), [verifytxoutproof\(\)](#)

gettxoutsetinfo *RPC-JSON API: gettxoutsetinfo*

Description

Returns statistics about the unspent transaction output set. Note this call may take some time.

Usage

```
gettxoutsetinfo(con)
```

Arguments

con object of class CONRPC.

Value

A S4-object of class ANSRPC.

Author(s)

Bernhard Pfaff

References

<https://bitcoin.org/en/developer-reference#gettxoutsetinfo>, <https://bitcoin.org/en/developer-reference#remote-procedure-calls-rpcs>

See Also

Other Blockchain RPCs: [decodescript\(\)](#), [getbestblockhash\(\)](#), [getblockchaininfo\(\)](#), [getblockcount\(\)](#), [getblockhash\(\)](#), [getblockheader\(\)](#), [getblock\(\)](#), [getchaintips\(\)](#), [getchaintxstats\(\)](#), [getdifficulty\(\)](#), [getmempoolancestors\(\)](#), [getmempooldescendants\(\)](#), [getmempoolentry\(\)](#), [getmempoolinfo\(\)](#), [getrawmempool\(\)](#), [gettxoutproof\(\)](#), [gettxout\(\)](#), [pruneblockchain\(\)](#), [verifychain\(\)](#), [verifytxoutproof\(\)](#)

getwalletinfo *RPC-JSON API: getwalletinfo*

Description

Returning information about bitcoin wallet.

Usage

```
getwalletinfo(con)
```

Arguments

con object of class CONRPC.

Value

A S4-object of class ANSRPC.

Author(s)

Bernhard Pfaff

References

<https://bitcoin.org/en/developer-reference#getwalletinfo>, <https://bitcoin.org/en/developer-reference#remote-procedure-calls-rpcs>

See Also

Other Control RPCs: [gethelp\(\)](#), [getinfo\(\)](#)

hash160

BCH hash160

Description

This function returns the hash by applying the sha256 hashing first and then to the resulting hash the ripemd160 algorithm.

Usage

```
hash160(d)
```

Arguments

d raw, vector.

Value

character, the value of d hashed with sha256 and ripemd160.

Author(s)

Bernhard Pfaff

References

<https://en.bitcoin.it/wiki/Address>

See Also

Other BchAddresses: [BCHADR-class](#), [BTCADR-class](#), [PubHash2BchAdr\(\)](#), [PubKey2PubHash\(\)](#), [base58CheckDecode\(\)](#), [base58CheckEncode\(\)](#), [concatHex\(\)](#), [decodeHex\(\)](#), [hash256\(\)](#), [validBchAdr\(\)](#), [validBtcAdr\(\)](#)

hash256

BCH hash256

Description

This function returns the hash by applying the sha256 hashing algorithm twice to a raw object.

Usage

```
hash256(d)
```

Arguments

d raw, vector.

Value

character, the value of d hashed twice.

Author(s)

Bernhard Pfaff

References

<https://en.bitcoin.it/wiki/Address>

See Also

Other BchAddresses: [BCHADR-class](#), [BTCADR-class](#), [PubHash2BchAdr\(\)](#), [PubKey2PubHash\(\)](#), [base58CheckDecode\(\)](#), [base58CheckEncode\(\)](#), [concatHex\(\)](#), [decodeHex\(\)](#), [hash160\(\)](#), [validBchAdr\(\)](#), [validBtcAdr\(\)](#)

int2date	<i>Convert time stamp to POSIX</i>
----------	------------------------------------

Description

This function returns the associated POSIXct time to the time stamp integer in a block header.

Usage

```
int2date(x)
```

Arguments

x integer, the block header time stamp

Value

An object of class POSIXct, POSIXt

Author(s)

Bernhard Pfaff

References

https://en.bitcoin.it/wiki/Block_timestamp

See Also

Other UtilityFuncs: [bkfee\(\)](#), [blockattime\(\)](#), [blockstats\(\)](#), [date2int\(\)](#), [intMaxDay\(\)](#), [intMinDay\(\)](#), [intRangeDay\(\)](#), [intRangePeriod\(\)](#), [timeofblock\(\)](#), [txfee\(\)](#), [txids\(\)](#), [txinids\(\)](#), [txstats\(\)](#), [utxoage\(\)](#), [utxotype\(\)](#), [utxovalue\(\)](#)

Examples

```
ts <- 1532954868
int2date(ts)
```

intMaxDay	<i>Integer representation of a day-end</i>
-----------	--

Description

This function returns the associated integer time for the end of a specific day (*i.e.*, 23:59:59 time).

Usage

```
intMaxDay(x)
```

Arguments

x POSIXct, date/time object.

Value

integer

Author(s)

Bernhard Pfaff

See Also

Other UtilityFuncs: [bkfee\(\)](#), [blockattime\(\)](#), [blockstats\(\)](#), [date2int\(\)](#), [int2date\(\)](#), [intMinDay\(\)](#), [intRangeDay\(\)](#), [intRangePeriod\(\)](#), [timeofblock\(\)](#), [txfee\(\)](#), [txids\(\)](#), [txinids\(\)](#), [txstats\(\)](#), [utxoage\(\)](#), [utxotype\(\)](#), [utxovalue\(\)](#)

Examples

```
d1 <- "2017-03-15"  
d1 <- intMaxDay(d1)  
d2 <- "2017-03-15 23:59:59"  
d2 <- intMaxDay(d2)  
identical(d1,d2)
```

intMinDay	<i>Integer representation of a day-begin</i>
-----------	--

Description

This function returns the associated integer time for the start of a specific day (*i.e.*, 00:00:00 time).

Usage

```
intMinDay(x)
```

Arguments

x POSIXct, date/time object.

Value

integer

Author(s)

Bernhard Pfaff

See Also

Other UtilityFuncs: [bkfee\(\)](#), [blockattime\(\)](#), [blockstats\(\)](#), [date2int\(\)](#), [int2date\(\)](#), [intMaxDay\(\)](#), [intRangeDay\(\)](#), [intRangePeriod\(\)](#), [timeofblock\(\)](#), [txfee\(\)](#), [txids\(\)](#), [txinids\(\)](#), [txstats\(\)](#), [utxoage\(\)](#), [utxotype\(\)](#), [utxovalue\(\)](#)

Examples

```
d1 <- "2017-03-15"  
d1 <- intMinDay(d1)  
d2 <- "2017-03-15 00:00:00"  
d2 <- intMinDay(d2)  
identical(d1,d2)
```

intRangeDay	<i>Integer range within a day</i>
-------------	-----------------------------------

Description

This function returns the associated integer times for the start and end of a specific day.

Usage

```
intRangeDay(x)
```

Arguments

x POSIXct, date/time object.

Value

integer

Author(s)

Bernhard Pfaff

See Also

Other UtilityFuncs: [bkfee\(\)](#), [blockatime\(\)](#), [blockstats\(\)](#), [date2int\(\)](#), [int2date\(\)](#), [intMaxDay\(\)](#), [intMinDay\(\)](#), [intRangePeriod\(\)](#), [timeofblock\(\)](#), [txfee\(\)](#), [txids\(\)](#), [txinids\(\)](#), [txstats\(\)](#), [utxoage\(\)](#), [utxotype\(\)](#), [utxovalue\(\)](#)

Examples

```
d1 <- "2017-03-15"  
intRangeDay(d1)  
intMinDay(d1)  
intMaxDay(d1)
```

intRangePeriod	<i>Integer range between two dates</i>
----------------	--

Description

This function returns the associated integer times for the start of date d1 and the end of date d2.

Usage

```
intRangePeriod(d1, d2)
```


Arguments

d1 POSIXct, date/time object.
d2 POSIXct, date/time object.

Value

integer

Author(s)

Bernhard Pfaff

See Also

Other UtilityFuncs: [bkfee\(\)](#), [blockatime\(\)](#), [blockstats\(\)](#), [date2int\(\)](#), [int2date\(\)](#), [intMaxDay\(\)](#), [intMinDay\(\)](#), [intRangeDay\(\)](#), [timeofblock\(\)](#), [txfee\(\)](#), [txids\(\)](#), [txinids\(\)](#), [txstats\(\)](#), [utxoage\(\)](#), [utxotype\(\)](#), [utxovalue\(\)](#)

Examples

```
d1 <- "2017-03-15"  
d2 <- "2017-04-15"  
intRangePeriod(d1, d2)  
intMinDay(d1)  
intMaxDay(d2)
```

isNull *Test for empty EC point*

Description

Checks whether an EC point does exist.

Usage

```
isNull(x)  
  
## S4 method for signature 'ECPOINT'  
isNull(x)
```

Arguments

x object

Value

logical

Author(s)

Bernhard Pfaff

References

<https://en.bitcoin.it/wiki/Secp256k1>

See Also

Other EllipticCurve: [ECPARAM-class](#), [ECPOINT-class](#), [EcpamOrNull-class](#), [containsPoint\(\)](#), [ecoperators](#), [ecparam\(\)](#), [ecpoint\(\)](#)

listbanned

RPC-JSON API: listbanned

Description

List all banned IPs/Subnets.

Usage

```
listbanned(con)
```

Arguments

con object of class CONRPC.

Value

A S4-object of class ANSRPC.

Author(s)

Bernhard Pfaff

References

<https://bitcoin.org/en/developer-reference#listbanned>, <https://bitcoin.org/en/developer-reference#remote-procedure-calls-rpcs>

See Also

Other Network RPCs: [addnode\(\)](#), [clearbanned\(\)](#), [disconnectnode\(\)](#), [getaddednodeinfo\(\)](#), [getconnectioncount\(\)](#), [getnettotals\(\)](#), [getnetworkinfo\(\)](#), [getpeerinfo\(\)](#), [ping\(\)](#)

NullOrCharacter-class *S4 Class Union NULL or character*

Description

S4-class union of NULL or character.

See Also

Other bitcoind functions: [ANSRPC-class](#), [CONRPC-class](#), [NullOrInteger-class](#), [conrpc\(\)](#), [rpcpost\(\)](#), [startbch\(\)](#), [startbtc\(\)](#), [stopbch\(\)](#), [stopbtc\(\)](#)

NullOrInteger-class *S4 Class Union NULL or integer*

Description

S4-class union of NULL or integer.

See Also

Other bitcoind functions: [ANSRPC-class](#), [CONRPC-class](#), [NullOrCharacter-class](#), [conrpc\(\)](#), [rpcpost\(\)](#), [startbch\(\)](#), [startbtc\(\)](#), [stopbch\(\)](#), [stopbtc\(\)](#)

ping *RPC-JSON API: ping*

Description

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.

Usage

ping(con)

Arguments

con object of class CONRPC.

Value

A S4-object of class ANSRPC.

Author(s)

Bernhard Pfaff

References

<https://bitcoin.org/en/developer-reference#ping>, <https://bitcoin.org/en/developer-reference#remote-procedure-calls-rpcs>

See Also

Other Network RPCs: [addnode\(\)](#), [clearbanned\(\)](#), [disconnectnode\(\)](#), [getaddednodeinfo\(\)](#), [getconnectioncount\(\)](#), [getnettotals\(\)](#), [getnetworkinfo\(\)](#), [getpeerinfo\(\)](#), [listbanned\(\)](#)

pruneblockchain

RPC-JSON API: pruneblockchain

Description

Pruning of blockchain.

Usage

```
pruneblockchain(con, height)
```

Arguments

con	object of class CONRPC.
height	integer The block height to prune up to.

Value

A S4-object of class ANSRPC.

Details

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.

Author(s)

Bernhard Pfaff

References

<https://bitcoin.org/en/developer-reference#pruneblockchain>, <https://bitcoin.org/en/developer-reference#remote-procedure-calls-rpcs>

See Also

Other Blockchain RPCs: [decodescript\(\)](#), [getbestblockhash\(\)](#), [getblockchaininfo\(\)](#), [getblockcount\(\)](#), [getblockhash\(\)](#), [getblockheader\(\)](#), [getblock\(\)](#), [getchaintips\(\)](#), [getchaintxstats\(\)](#), [getdifficulty\(\)](#), [getmempoolancestors\(\)](#), [getmempooldescendants\(\)](#), [getmempoolentry\(\)](#), [getmempoolinfo\(\)](#), [getrawmempool\(\)](#), [gettxoutproof\(\)](#), [gettxoutsetinfo\(\)](#), [gettxout\(\)](#), [verifychain\(\)](#), [verifytxoutproof\(\)](#)

PubHash2BchAdr

Create BCH address from public key hash

Description

This function returns the corresponding BCH address from a hashed public key.

Usage

PubHash2BchAdr(pubhash)

Arguments

pubhash character, the public key hash.

Value

character, the BCH address

Author(s)

Bernhard Pfaff

References

<https://en.bitcoin.it/wiki/Address>

See Also

Other BchAddresses: [BCHADR-class](#), [BTCADR-class](#), [PubKey2PubHash\(\)](#), [base58CheckDecode\(\)](#), [base58CheckEncode\(\)](#), [concatHex\(\)](#), [decodeHex\(\)](#), [hash160\(\)](#), [hash256\(\)](#), [validBchAdr\(\)](#), [validBtcAdr\(\)](#)

PubHash2BtcAdr	<i>Create BCH address from public key hash (BTC alias)</i>
----------------	--

Description

This function returns the corresponding BTC address from a hashed public key.

Usage

```
PubHash2BtcAdr(pubhash)
```

Arguments

pubhash character, the public key hash.

Value

character, the BTC address

Author(s)

Bernhard Pfaff

References

<https://en.bitcoin.it/wiki/Address>

PubKey2PubHash	<i>Create public key hash from 512-bit public key</i>
----------------	---

Description

This function returns the associated public key hash from a 512-bit public key by using the hash160() function.

Usage

```
PubKey2PubHash(pubkey, mainnet = TRUE)
```

Arguments

pubkey character, the public key.
mainnet logical, whether the key should correspond to the mainnet or testnet.

Value

character, the hash of a public key

Author(s)

Bernhard Pfaff

References

<https://en.bitcoin.it/wiki/Address>

See Also

Other BchAddresses: [BCHADR-class](#), [BTCADR-class](#), [PubHash2BchAdr\(\)](#), [base58CheckDecode\(\)](#), [base58CheckEncode\(\)](#), [concatHex\(\)](#), [decodeHex\(\)](#), [hash160\(\)](#), [hash256\(\)](#), [validBchAdr\(\)](#), [validBtcAdr\(\)](#)

rpcpost

HTTP post of RPC-JSON

Description

This function executes an RPC-JSON post.

Usage

```
rpcpost(con, api, plist = list())
```

Arguments

con	CONRPC object, returned from <code>conrpc()</code> .
api	character the name of the RPC function.
plist	list a named list object of the parameters for api

Value

A list object, coerced JSON answer from RPC.

Author(s)

Bernhard Pfaff

See Also

Other bitcoind functions: [ANSRPC-class](#), [CONRPC-class](#), [NullOrCharacter-class](#), [NullOrInteger-class](#), [conrpc\(\)](#), [startbch\(\)](#), [startbtc\(\)](#), [stopbch\(\)](#), [stopbtc\(\)](#)

show	<i>show-methods</i>
------	---------------------

Description

Defined show-methods for S4-classes.

Usage

```
## S4 method for signature 'ANSRPC'  
show(object)  
  
## S4 method for signature 'BCHADR'  
show(object)  
  
## S4 method for signature 'BTCADR'  
show(object)  
  
## S4 method for signature 'ECPARAM'  
show(object)
```

Arguments

object a S4-class object.

startbch	<i>Start bitcoind server process</i>
----------	--------------------------------------

Description

This function does start the bitcoind-server process. It should only be called when no suitable RPC-JSON process is running

Usage

```
startbch(confbch)
```

Arguments

confbch CONRPC object, returned from conrpc().

Details

The process is started by calling `system()`. Hereby, the options: `rpcuser`, `rpcpassword` and `conf` are used in the call to `bitcoind`.

Value

NULL

Author(s)

Bernhard Pfaff

See Also

Other bitcoind functions: [ANSRPC-class](#), [CONRPC-class](#), [NullOrCharacter-class](#), [NullOrInteger-class](#), [conrpc\(\)](#), [rpcpost\(\)](#), [startbtc\(\)](#), [stopbch\(\)](#), [stopbtc\(\)](#)

`startbtc`*Start bitcoind server process (BTC alias)*

Description

This function is an alias of startbch

Usage`startbtc(confbtc)`**Arguments**

`confbtc` CONRPC object, returned from `conrpc()`.

Details

The process is started by calling `system()`. Hereby, the options: `rpcuser`, `rpcpassword` and `conf` are used in the call to `bitcoind`.

Value

NULL

Author(s)

Bernhard Pfaff

See Also

Other bitcoind functions: [ANSRPC-class](#), [CONRPC-class](#), [NullOrCharacter-class](#), [NullOrInteger-class](#), [conrpc\(\)](#), [rpcpost\(\)](#), [startbch\(\)](#), [stopbch\(\)](#), [stopbtc\(\)](#)

stopbch	<i>Stop bitcoind server process</i>
---------	-------------------------------------

Description

This function stops a running bitcoind process. It calls bitcoin-cli stop *via* the R function `system()`.

Usage

```
stopbch(confbch)
```

Arguments

confbch CONRPC object, returned from `conrpc()`.

Author(s)

Bernhard Pfaff

See Also

Other bitcoind functions: [ANSRPC-class](#), [CONRPC-class](#), [NullOrCharacter-class](#), [NullOrInteger-class](#), [conrpc\(\)](#), [rpcpost\(\)](#), [startbch\(\)](#), [startbtc\(\)](#), [stopbtc\(\)](#)

stopbtc	<i>Stop bitcoind server process (BTC alias)</i>
---------	---

Description

This function stops a running bitcoind process. It calls bitcoin-cli stop *via* the R function `system()`.

Usage

```
stopbtc(confbtc)
```

Arguments

confbtc CONRPC object, returned from `conrpc()`.

Author(s)

Bernhard Pfaff

See Also

Other bitcoind functions: [ANSRPC-class](#), [CONRPC-class](#), [NullOrCharacter-class](#), [NullOrInteger-class](#), [conrpc\(\)](#), [rpcpost\(\)](#), [startbch\(\)](#), [startbtc\(\)](#), [stopbch\(\)](#)

timeofblock	<i>Time of a block</i>
-------------	------------------------

Description

This function returns the time of a block in GMT.

Usage

```
timeofblock(con, height)
```

Arguments

con	CONRPC, configuration object.
height	integer, the height of the block.

Value

POSIXct

Author(s)

Bernhard Pfaff

See Also

Other UtilityFuncs: [bkfee\(\)](#), [blockattime\(\)](#), [blockstats\(\)](#), [date2int\(\)](#), [int2date\(\)](#), [intMaxDay\(\)](#), [intMinDay\(\)](#), [intRangeDay\(\)](#), [intRangePeriod\(\)](#), [txfee\(\)](#), [txids\(\)](#), [txinids\(\)](#), [txstats\(\)](#), [utxoage\(\)](#), [utxotype\(\)](#), [utxovalue\(\)](#)

txfee	<i>Compute fee of a transaction</i>
-------	-------------------------------------

Description

This function returns the implicit fee of a transaction, by computing the difference between the sum of its inputs and the sum of its outputs.

Usage

```
txfee(con, txid)
```

Arguments

con	CONRPC, configuration object.
txid	character, the id of the transaction.

Value

numeric

Author(s)

Bernhard Pfaff

See Also

Other UtilityFuncs: [bkfee\(\)](#), [blockattime\(\)](#), [blockstats\(\)](#), [date2int\(\)](#), [int2date\(\)](#), [intMaxDay\(\)](#), [intMinDay\(\)](#), [intRangeDay\(\)](#), [intRangePeriod\(\)](#), [timeofblock\(\)](#), [txids\(\)](#), [txinids\(\)](#), [txstats\(\)](#), [utxoage\(\)](#), [utxotype\(\)](#), [utxovalue\(\)](#)

txids	<i>Retrieve TX Ids in block</i>
-------	---------------------------------

Description

This function retrieves the transaction IDs in a block.

Usage

```
txids(con, height, excoinbase = TRUE)
```

Arguments

con	CONRPC, configuration object.
height	integer, the block's height.
excoinbase	logical, whether coinbase transaction should be excluded (default is TRUE).

Value

character

Author(s)

Bernhard Pfaff

See Also

Other UtilityFuncs: [bkfee\(\)](#), [blockattime\(\)](#), [blockstats\(\)](#), [date2int\(\)](#), [int2date\(\)](#), [intMaxDay\(\)](#), [intMinDay\(\)](#), [intRangeDay\(\)](#), [intRangePeriod\(\)](#), [timeofblock\(\)](#), [txfee\(\)](#), [txinids\(\)](#), [txstats\(\)](#), [utxoage\(\)](#), [utxotype\(\)](#), [utxovalue\(\)](#)

txinids

Retrieving the input transaction IDs

Description

This function returns the transaction IDs of the inputs for a given transaction.

Usage

```
txinids(con, txid)
```

Arguments

con	CONRPC, configuration object.
txid	character, the id of the transaction.

Value

data.frame, the transaction ID(s) and the position(s) of the previous UTXO(s).

Author(s)

Bernhard Pfaff

See Also

Other UtilityFuncs: [bkfee\(\)](#), [blockattime\(\)](#), [blockstats\(\)](#), [date2int\(\)](#), [int2date\(\)](#), [intMaxDay\(\)](#), [intMinDay\(\)](#), [intRangeDay\(\)](#), [intRangePeriod\(\)](#), [timeofblock\(\)](#), [txfee\(\)](#), [txids\(\)](#), [txstats\(\)](#), [utxoage\(\)](#), [utxotype\(\)](#), [utxovalue\(\)](#)

txstats	<i>Statistics of a transaction</i>
---------	------------------------------------

Description

This function returns key statistics/characteristics of a transaction.

Usage

```
txstats(con, txid)
```

Arguments

con	CONRPC, configuration object.
txid	character, the id of the transaction.

Value

data.frame

Author(s)

Bernhard Pfaff

See Also

Other UtilityFuncs: [bkfee\(\)](#), [blockatime\(\)](#), [blockstats\(\)](#), [date2int\(\)](#), [int2date\(\)](#), [intMaxDay\(\)](#), [intMinDay\(\)](#), [intRangeDay\(\)](#), [intRangePeriod\(\)](#), [timeofblock\(\)](#), [txfee\(\)](#), [txids\(\)](#), [txinids\(\)](#), [utxoage\(\)](#), [utxotype\(\)](#), [utxovalue\(\)](#)

utxoage	<i>Age of UTXOs</i>
---------	---------------------

Description

This function returns a `difftime` object measuring the elapsed time(s) between the UTXO(s) in a transaction and its input(s) (previous UTXO(s)).

Usage

```
utxoage(con, txid, units = c("auto", "secs", "mins", "hours", "days", "weeks"))
```

Arguments

con	CONRPC, configuration object.
txid	character, the id of the transaction.
units	character, the time difference units; passed to <code>difftime()</code> .

Value

difftime

Author(s)

Bernhard Pfaff

See Also

Other UtilityFuncs: [bkfee\(\)](#), [blockattime\(\)](#), [blockstats\(\)](#), [date2int\(\)](#), [int2date\(\)](#), [intMaxDay\(\)](#), [intMinDay\(\)](#), [intRangeDay\(\)](#), [intRangePeriod\(\)](#), [timeofblock\(\)](#), [txfee\(\)](#), [txids\(\)](#), [txinids\(\)](#), [txstats\(\)](#), [utxotype\(\)](#), [utxovalue\(\)](#)

utxotype

Retrieving types of UTXOs

Description

This function returns the types of the UTXO(s) in a transaction.

Usage

```
utxotype(con, txid)
```

Arguments

con	CONRPC, configuration object.
txid	character, the id of the transaction.

Value

character

Author(s)

Bernhard Pfaff

See Also

Other UtilityFuncs: [bkfee\(\)](#), [blockattime\(\)](#), [blockstats\(\)](#), [date2int\(\)](#), [int2date\(\)](#), [intMaxDay\(\)](#), [intMinDay\(\)](#), [intRangeDay\(\)](#), [intRangePeriod\(\)](#), [timeofblock\(\)](#), [txfee\(\)](#), [txids\(\)](#), [txinids\(\)](#), [txstats\(\)](#), [utxoage\(\)](#), [utxovalue\(\)](#)

utxovalue	<i>Retrieving values of UTXOs</i>
-----------	-----------------------------------

Description

This function returns the values of UTXO(s) in a transaction.

Usage

```
utxovalue(con, txid)
```

Arguments

con	CONRPC, configuration object.
txid	character, the id of the transaction.

Value

numeric

Author(s)

Bernhard Pfaff

See Also

Other UtilityFuncs: [bkfee\(\)](#), [blockattime\(\)](#), [blockstats\(\)](#), [date2int\(\)](#), [int2date\(\)](#), [intMaxDay\(\)](#), [intMinDay\(\)](#), [intRangeDay\(\)](#), [intRangePeriod\(\)](#), [timeofblock\(\)](#), [txfee\(\)](#), [txids\(\)](#), [txinids\(\)](#), [txstats\(\)](#), [utxoage\(\)](#), [utxotype\(\)](#)

validBchAdr	<i>Validate S4-class BCHADR</i>
-------------	---------------------------------

Description

This function validates objects of S4-class BCHADR. Hereby, checks are conducted with respect to the first character of the addresses; their consistency with the net version and the correspondence of the checksums.

Usage

```
validBchAdr(object)
```

Arguments

object	BCHADR object
--------	---------------

Author(s)

Bernhard Pfaff

References

<https://en.bitcoin.it/wiki/Address>

See Also

Other BchAddresses: [BCHADR-class](#), [BTCADR-class](#), [PubHash2BchAdr\(\)](#), [PubKey2PubHash\(\)](#), [base58CheckDecode\(\)](#), [base58CheckEncode\(\)](#), [concatHex\(\)](#), [decodeHex\(\)](#), [hash160\(\)](#), [hash256\(\)](#), [validBtcAdr\(\)](#)

validBtcAdr

Validate S4-class BTCADR (BTC alias)

Description

This function validates objects of S4-class BTCADR. Hereby, checks are conducted with respect to the first character of the addresses; their consistency with the net version and the correspondence of the checksums.

Usage

```
validBtcAdr(object)
```

Arguments

object BTCADR object

Author(s)

Bernhard Pfaff

References

<https://en.bitcoin.it/wiki/Address>

See Also

Other BchAddresses: [BCHADR-class](#), [BTCADR-class](#), [PubHash2BchAdr\(\)](#), [PubKey2PubHash\(\)](#), [base58CheckDecode\(\)](#), [base58CheckEncode\(\)](#), [concatHex\(\)](#), [decodeHex\(\)](#), [hash160\(\)](#), [hash256\(\)](#), [validBchAdr\(\)](#)

verifychain	<i>RPC-JSON API: verifychain</i>
-------------	----------------------------------

Description

Verifies blockchain database.

Usage

```
verifychain(con, checklevel = NULL, nblocks = NULL)
```

Arguments

con	object of class CONRPC.
checklevel	integer (optional, 0-4, default=3), how thorough the block verification is.
nblocks	integer (optional, default=6, 0=all), the number of blocks to check.

Value

A S4-object of class ANSRPC.

Author(s)

Bernhard Pfaff

References

<https://bitcoin.org/en/developer-reference#verifychain>, <https://bitcoin.org/en/developer-reference#remote-procedure-calls-rpcs>

See Also

Other Blockchain RPCs: [decodescript\(\)](#), [getbestblockhash\(\)](#), [getblockchaininfo\(\)](#), [getblockcount\(\)](#), [getblockhash\(\)](#), [getblockheader\(\)](#), [getblock\(\)](#), [getchaintips\(\)](#), [getchaintxstats\(\)](#), [getdifficulty\(\)](#), [getmempoolancestors\(\)](#), [getmempooldescendants\(\)](#), [getmempoolentry\(\)](#), [getmempoolinfo\(\)](#), [getrawmempool\(\)](#), [gettxoutproof\(\)](#), [gettxoutsetinfo\(\)](#), [gettxout\(\)](#), [pruneblockchain\(\)](#), [verifytxoutproof\(\)](#)

verifytxoutproof *RPC-JSON API: verifytxoutproof*

Description

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.

Usage

```
verifytxoutproof(con, proof)
```

Arguments

con	object of class CONRPC.
proof	character the hex-encoded proof generated by <code>gettxoutproof</code> .

Value

A S4-object of class ANSRPC.

Author(s)

Bernhard Pfaff

References

<https://bitcoin.org/en/developer-reference#verifytxoutproof>, <https://bitcoin.org/en/developer-reference#remote-procedure-calls-rpcs>

See Also

Other Blockchain RPCs: [decodescript\(\)](#), [getbestblockhash\(\)](#), [getblockchaininfo\(\)](#), [getblockcount\(\)](#), [getblockhash\(\)](#), [getblockheader\(\)](#), [getblock\(\)](#), [getchaintips\(\)](#), [getchaintxstats\(\)](#), [getdifficulty\(\)](#), [getmempoolancestors\(\)](#), [getmempooldescendants\(\)](#), [getmempoolentry\(\)](#), [getmempoolinfo\(\)](#), [getrawmempool\(\)](#), [gettxoutproof\(\)](#), [gettxoutsetinfo\(\)](#), [gettxout\(\)](#), [pruneblockchain\(\)](#), [verifychain\(\)](#)

Index

* **BchAddresses**

- base58CheckDecode, 5
- base58CheckEncode, 5
- BCHADR-class, 6
- BTCADR-class, 9
- concatHex, 10
- decodeHex, 14
- hash160, 43
- hash256, 44
- PubHash2BchAdr, 53
- PubKey2PubHash, 54
- validBchAdr, 64
- validBtcAdr, 65

* **Blockchain RPCs**

- decodescript, 15
- getbestblockhash, 23
- getblock, 23
- getblockchaininfo, 24
- getblockcount, 25
- getblockhash, 26
- getblockheader, 27
- getchaintips, 28
- getchaintxstats, 28
- getdifficulty, 30
- getmempoolancestors, 32
- getmempooldescendants, 33
- getmempoolentry, 34
- getmempoolinfo, 35
- getrawmempool, 38
- gettxout, 40
- gettxoutproof, 41
- gettxoutsetinfo, 42
- pruneblockchain, 52
- verifychain, 66
- verifytxoutproof, 67

* **BtcAddresses**

- PubHash2BtcAdr, 54

* **Control RPCs**

- gethelp, 31

- getinfo, 31

- getwalletinfo, 42

* **EllipticCurve**

- containsPoint, 12
- ecoperators, 17
- ecparam, 18
- ECPARAM-class, 19
- EcparamOrNull-class, 20
- ecpoint, 20
- ECPOINT-class, 21
- isNull, 49

* **Network RPCs**

- addnode, 3
- clearbanned, 10
- disconnectnode, 16
- getaddednodeinfo, 22
- getconnectioncount, 29
- getnettotals, 35
- getnetworkinfo, 36
- getpeerinfo, 37
- listbanned, 50
- ping, 51

* **RawTransactions RPCs**

- decoderawtransaction, 15
- getrawtransaction, 39

* **UtilityFuncs**

- bkfee, 7
- blockattime, 8
- blockstats, 8
- date2int, 13
- int2date, 45
- intMaxDay, 46
- intMinDay, 47
- intRangeDay, 48
- intRangePeriod, 48
- timeofblock, 59
- txfee, 60
- txids, 60
- txinids, 61

- txstats, 62
- utxoage, 62
- utxotype, 63
- utxovalue, 64
- * **bitcoind functions**
 - ANSRPC-class, 4
 - conrpc, 11
 - CONRPC-class, 12
 - NullOrCharacter-class, 51
 - NullOrInteger-class, 51
 - rpcpost, 55
 - startbch, 56
 - startbtc, 57
 - stopbch, 58
 - stopbtc, 58
- *, ECPOINT, bigz-method (ecoperators), 17
- *, bigz, ECPOINT-method (ecoperators), 17
- +, ECPOINT, ECPOINT-method (ecoperators), 17
- addnode, 3, 10, 17, 22, 30, 36, 37, 50, 52
- AND (ecoperators), 17
- AND, bigz, bigz-method (ecoperators), 17
- ANSRPC-class, 4
- base58CheckDecode, 5, 6, 7, 9, 11, 14, 44, 53, 55, 65
- base58CheckEncode, 5, 5, 7, 9, 11, 14, 44, 53, 55, 65
- BCHADR-class, 6
- bkfee, 7, 8, 9, 14, 45–49, 59–64
- blockattime, 7, 8, 9, 14, 45–49, 59–64
- blockstats, 7, 8, 8, 14, 45–49, 59–64
- BTCADR-class, 9
- clearbanned, 4, 10, 17, 22, 30, 36, 37, 50, 52
- concatHex, 5–7, 9, 10, 14, 44, 53, 55, 65
- conrpc, 4, 11, 12, 51, 55, 57–59
- CONRPC-class, 12
- containsPoint, 12, 18–22, 50
- containsPoint, ECPARAM, bigz, bigz-method (containsPoint), 12
- containsPoint, ECPARAM, character, character-method (containsPoint), 12
- containsPoint, ECPARAM, integer, integer-method (containsPoint), 12
- date2int, 7–9, 13, 45–49, 59–64
- decodeHex, 5–7, 9, 11, 14, 44, 53, 55, 65
- decoderawtransaction, 15, 39
- decodescript, 15, 23–30, 33–35, 38, 40–42, 53, 66, 67
- disconnectnode, 4, 10, 16, 22, 30, 36, 37, 50, 52
- doubleUp (ecoperators), 17
- doubleUp, ECPOINT-method (ecoperators), 17
- ecoperators, 13, 17, 19–22, 50
- ecparam, 13, 18, 18, 20–22, 50
- ECPARAM-class, 19
- EcparamOrNull-class, 20
- ecpoint, 13, 18–20, 20, 22, 50
- ECPOINT-class, 21
- getaddednodeinfo, 4, 10, 17, 22, 30, 36, 37, 50, 52
- getbestblockhash, 16, 23, 24–30, 33–35, 38, 40–42, 53, 66, 67
- getblock, 16, 23, 23, 25–30, 33–35, 38, 40–42, 53, 66, 67
- getblockchaininfo, 16, 23, 24, 24, 25–30, 33–35, 38, 40–42, 53, 66, 67
- getblockcount, 16, 23–25, 25, 26–30, 33–35, 38, 40–42, 53, 66, 67
- getblockhash, 16, 23–25, 26, 27–30, 33–35, 38, 40–42, 53, 66, 67
- getblockheader, 16, 23–26, 27, 28–30, 33–35, 38, 40–42, 53, 66, 67
- getchaintips, 16, 23–27, 28, 29, 30, 33–35, 38, 40–42, 53, 66, 67
- getchaintxstats, 16, 23–28, 28, 30, 33–35, 38, 40–42, 53, 66, 67
- getconnectioncount, 4, 10, 17, 22, 29, 36, 37, 50, 52
- getdifficulty, 16, 23–29, 30, 33–35, 38, 40–42, 53, 66, 67
- gethelp, 31, 32, 43
- getinfo, 31, 31, 43
- getmempoolancestors, 16, 23–30, 32, 33–35, 38, 40–42, 53, 66, 67
- getmempooldescendants, 16, 23–30, 33, 33, 34, 35, 38, 40–42, 53, 66, 67
- getmempoolentry, 16, 23–30, 33, 34, 35, 38, 40–42, 53, 66, 67
- getmempoolinfo, 16, 23–30, 33, 34, 35, 38, 40–42, 53, 66, 67
- getnettotals, 4, 10, 17, 22, 30, 35, 37, 50, 52

- getnetworkinfo, [4](#), [10](#), [17](#), [22](#), [30](#), [36](#), [36](#), [37](#), [50](#), [52](#)
- getpeerinfo, [4](#), [10](#), [17](#), [22](#), [30](#), [36](#), [37](#), [37](#), [50](#), [52](#)
- getrawmempool, [16](#), [23–30](#), [33–35](#), [38](#), [40–42](#), [53](#), [66](#), [67](#)
- getrawtransaction, [15](#), [39](#)
- gettxout, [16](#), [23–30](#), [33–35](#), [38](#), [40](#), [41](#), [42](#), [53](#), [66](#), [67](#)
- gettxoutproof, [16](#), [23–30](#), [33–35](#), [38](#), [40](#), [41](#), [42](#), [53](#), [66](#), [67](#)
- gettxoutsetinfo, [16](#), [23–30](#), [33–35](#), [38](#), [40](#), [41](#), [42](#), [53](#), [66](#), [67](#)
- getwalletinfo, [31](#), [32](#), [42](#)
- hash160, [5–7](#), [9](#), [11](#), [14](#), [43](#), [44](#), [53](#), [55](#), [65](#)
- hash256, [5–7](#), [9](#), [11](#), [14](#), [44](#), [44](#), [53](#), [55](#), [65](#)
- int2date, [7–9](#), [14](#), [45](#), [46–49](#), [59–64](#)
- intMaxDay, [7–9](#), [14](#), [45](#), [46](#), [47–49](#), [59–64](#)
- intMinDay, [7–9](#), [14](#), [45](#), [46](#), [47](#), [48](#), [49](#), [59–64](#)
- intRangeDay, [7–9](#), [14](#), [45–47](#), [48](#), [49](#), [59–64](#)
- intRangePeriod, [7–9](#), [14](#), [45–48](#), [48](#), [59–64](#)
- isNull, [13](#), [18–22](#), [49](#)
- isNull, ECPOINT-method (isNull), [49](#)
- leftmostBit (ecoperators), [17](#)
- leftmostBit, bigz-method (ecoperators), [17](#)
- listbanned, [4](#), [10](#), [17](#), [22](#), [30](#), [36](#), [37](#), [50](#), [52](#)
- NullOrCharacter-class, [51](#)
- NullOrInteger-class, [51](#)
- ping, [4](#), [10](#), [17](#), [22](#), [30](#), [36](#), [37](#), [50](#), [51](#)
- pruneblockchain, [16](#), [23–30](#), [33–35](#), [38](#), [40–42](#), [52](#), [66](#), [67](#)
- PubHash2BchAdr, [5–7](#), [9](#), [11](#), [14](#), [44](#), [53](#), [55](#), [65](#)
- PubHash2BtcAdr, [54](#)
- PubKey2PubHash, [5–7](#), [9](#), [11](#), [14](#), [44](#), [53](#), [54](#), [65](#)
- rpcpost, [4](#), [12](#), [51](#), [55](#), [57–59](#)
- show, [56](#)
- show, ANSRPC-method (show), [56](#)
- show, BCHADR-method (show), [56](#)
- show, BTCADR-method (show), [56](#)
- show, ECPARAM-method (show), [56](#)
- startbch, [4](#), [12](#), [51](#), [55](#), [56](#), [57–59](#)
- startbtc, [4](#), [12](#), [51](#), [55](#), [57](#), [58](#), [59](#)
- stopbch, [4](#), [12](#), [51](#), [55](#), [57](#), [58](#), [59](#)
- stopbtc, [4](#), [12](#), [51](#), [55](#), [57](#), [58](#), [58](#)
- timeofblock, [7–9](#), [14](#), [45–49](#), [59](#), [60–64](#)
- txfee, [7–9](#), [14](#), [45–49](#), [59](#), [60](#), [61–64](#)
- txids, [7–9](#), [14](#), [45–49](#), [59](#), [60](#), [60](#), [61–64](#)
- txinids, [7–9](#), [14](#), [45–49](#), [59–61](#), [61](#), [62–64](#)
- txstats, [7–9](#), [14](#), [45–49](#), [59–61](#), [62](#), [63](#), [64](#)
- utxoage, [7–9](#), [14](#), [45–49](#), [59–62](#), [62](#), [63](#), [64](#)
- utxotype, [7–9](#), [14](#), [45–49](#), [59–63](#), [63](#), [64](#)
- utxovalue, [7–9](#), [14](#), [45–49](#), [59–63](#), [64](#)
- validBchAdr, [5–7](#), [9](#), [11](#), [14](#), [44](#), [53](#), [55](#), [64](#), [65](#)
- validBtcAdr, [5–7](#), [9](#), [11](#), [14](#), [44](#), [53](#), [55](#), [65](#), [65](#)
- verifychain, [16](#), [23–30](#), [33–35](#), [38](#), [40–42](#), [53](#), [66](#), [67](#)
- verifytxoutproof, [16](#), [23–30](#), [33–35](#), [38](#), [40–42](#), [53](#), [66](#), [67](#)