I. Introduction

Zcash is an implementation of the Decentralized Anonymous Payment scheme Zerocash, with security fixes and improvements to performance and functionality. In this document I will talk about Zcash Payment API and ZCash block explorer API

II. Zcash Payment API

Zcash protocol extends the bitcoin Core API and use a new RPC calls to support private Zcash payments.
Zcash payments make use of two address formats:

- taddr - an address for transparent funds (just like a Bitcoin address, value stored in UTXOs)
- zaddr - an address for private funds (value stored in objects called notes) When transferring funds from one taddr to another taddr, you can use either the existing Bitcoin RPC calls or the new Zcash RPC calls. When a transfer involves zaddrs, you must use the new Zcash RPC calls.

1. Zcash API commands

RPC calls by category:

- Accounting: z_getbalance, z_gettotalbalance
- Addresses : z_getnewaddress, z_listaddresses, z_validateaddress
- Keys : z_exportkey, z_importkey, z_exportwallet, z_importwallet
- Operation: z_getoperationresult, z_getoperationstatus, z_listoperationids
- Payment : z_listreceivedbyaddress, z_sendmany, z_shieldcoinbase

RPC parameter conventions:

- taddr: Transparent address •
- zaddr : Private address •
- address: Accepts both private and transparent addresses.
- amount: JSON format double-precision number with 1 ZC expressed as 1.00000000.
- memo: Metadata expressed in hexadecimal format. Limited to 512 bytes, the current size of the memo field of a private transaction. Zero padding is automatic.

Accounting

Command | Parameters | Description — | — | — z_getbalance| address [minconf=1] | Returns the balance of a taddr or zaddr belonging to the node's wallet.Optionally set the minimum number of confirmations a private or transaction transaction must have in order to be included in the balance. Use 0 to count unconfirmed transactions. z_gettotalbalance| [minconf=1] | Return the total value of funds stored in the node's wallet.Optionally set the minimum number of confirmations a private or transparent transaction must have in order to be included in the balance. Use 0 to count unconfirmed transactions.Output:{"transparent" : 1.23,"private" : 4.56,"total" : 5.79} Addresses Command | Parameters | Description — | — | — z_getnewaddress | | Return a new zaddr for sending and receiving payments. The spending key for this zaddr will be added to the node's wallet.Output:zN68D8hSs3. . . z_listaddresses | | Returns a list of all the zaddrs in this node's wallet for which you have a spending key.Output:{ ["z123. . . ", "z456. . . ", "z789. . . "] } z_validateaddress | zaddr | Return information about a given zaddr.Output:{"isvalid" : true,"address" : "zcWsmq. . . ","payingkey": "f5bb3c. . . ","transmissionkey" : "7a58c7. . . ","ismine" : true} Key Management Command | Parameters | Description — | — | — z_exportkey | zaddr | Requires an unlocked wallet or an unencrypted wallet.Return a zkey for a given zaddr belonging to the node's wallet.The key will be returned as a string formatted using Base58Check as described in the Zcash protocol spec.Output:AKWUAkypwQjhZ6LLNaMuuuLcmZ6gt5UFyo8m3jGutvALmwZKLdR5 z_importkey |

zkey [rescan=true] | Wallet must be unlocked.Add a zkey as returned by z_exportkey to a node's wallet. The key should be formatted using Base58Check as described in the Zcash protocol spec. Set rescan to true (the default) to rescan the entire local block database for transactions affecting any address or pubkey script in the wallet (including transactions affecting the newly-added address for this spending key). z_exportwallet | filename | Requires an unlocked wallet or an unencrypted wallet. Creates or overwrites a file with taddr private keys and zaddr private keys in a human-readable format. Filename is the file in which the wallet dump will be placed. May be prefaced by an absolute file path. An existing file with that name will be overwritten. No value is returned but a JSON-RPC error will be reported if a failure occurred. z_importwallet | filename | Requires an unlocked wallet or an unencrypted wallet. Imports private keys from a file in wallet export file format (see z_exportwallet). These keys will be added to the keys currently in the wallet. This call may need to rescan all or parts of the block chain for transactions affecting the newly-added keys, which may take several minutes. Filename is the file to import. The path is relative to zcashd's working directory. No value is returned but a JSON-RPC error will be reported if a failure occurred.

Payment

Command | Parameters | Description — | — | — z_listreceivedbyaddress | zaddr [minconf=1] | Return a list of amounts received by a zaddr belonging to the node's wallet. Optionally set the minimum number of confirmations which a received amount must have in order to be included in the result. Use 0 to count unconfirmed transactions. Output: [{"txid": "4a0f. . . ",“ amount": 0.54,"memo":"F0FF. . . ",}, {. . . }, {. . . }] z_sendmany | fromaddress amounts [minconf=1] [fee=0.0001] | This is an Asynchronous RPC callSend funds from an address to multiple outputs. The address can be either a taddr or a zaddr. Amounts is a list containing key/value pairs corresponding to the addresses and amount to pay. Each output address can be in taddr or zaddr format. When sending to a zaddr, you also have the option of attaching a memo in hexadecimal format. NOTE: When sending coinbase funds to a zaddr, the node's wallet does not allow any change. Put another way, spending a partial amount of a coinbase utxo is not allowed. This is not a consensus rule but a local wallet rule due to the current implementation of z_sendmany. In future, this rule may be removed. Example of Outputs parameter: [{"address":"t123. . . ", "amount":0.005},,{"address":"z010. . . ","amount":0.03, "memo":"f508af. . . "}] Optionally set the minimum number of confirmations which a private or transparent transaction must have in order to be used as an input. When sending from a zaddr, minconf must be greater than zero. Optionally set a transaction fee, which by default is 0.0001 ZEC. Any transparent change will be sent to a new transparent address. Any private change will be sent back to the zaddr being used as the source of funds. Returns an operationid. You use the operationid value with z_getoperationstatus and z_getoperationresult to obtain the result of sending funds, which if successful, will be a txid. z_shieldcoinbase | fromaddress toaddress [fee=0.0001] [limit=50] | This is an Asynchronous RPC callShield transparent coinbase funds by sending to a shielded z address. Utxos selected for shielding will be locked. If there is an error, they are unlocked. The RPC call listlockunspent can be used to return a list of locked utxos. The number of coinbase utxos selected for shielding can be set with the limit parameter, which has a default value of 50. If the parameter is set to 0, the number of utxos selected is limited by the -mempooltxinputlimit option. Any limit is constrained by a consensus rule defining a maximum transaction size of 100000 bytes. the from address is a taddr or "*" for all taddrs belonging to the wallet. The to address is a zaddr. The default fee is 0.0001.Returns an object containing an operationid which can be used with z_getoperationstatus and z_getoperationresult, along with key-value pairs regarding how many utxos are being shielded in this transaction and what remains to be shielded.

Operations

Asynchronous calls return an OperationStatus object which is a JSON object with the following defined key-value pairs:

- operationid: unique identifier for the async operation. Use this value with z_getoperationstatus or z_getoperationresult to poll and query the operation and obtain its result.
- status: current status of operation
  – queued: operation is pending execution
  – executing: operation is currently being executed
  – cancelled
  – failed.
  – Success
- result : result object if the status is ‘success’. The exact form of the result object is dependent on the call itself.
- error: error object if the status is ‘failed’. The error object has the following key-value pairs:
  – code: number
  – message: error message
  Depending on the type of asynchronous call, there may be other key-value pairs. For example, a z_sendmany operation will also include the following in an OperationStatus object:
- method: name of operation e.g. z_sendmany •
- params : an object containing the parameters to z_sendmany

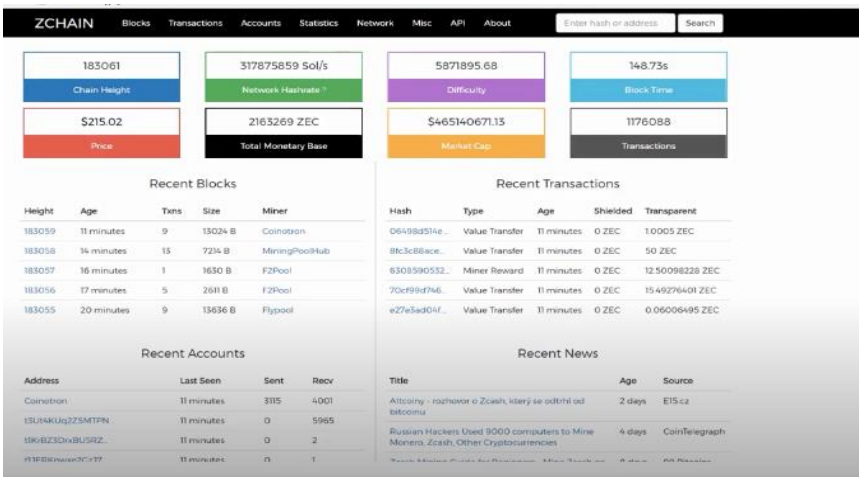
Currently, as soon as you retrieve the operation status for an operation which has finished, that is it has either succeeded, failed, or been cancelled, the operation and any associated information is removed. It is currently not possible to cancel operations.

Command | Parameters | Description — | — | — z_getoperationresult | [operationids] | Return OperationStatus JSON objects for all completed operations the node is currently aware of, and then remove the operation from memory. Operationids is an optional array to filter which operations you want to receive status objects for. Output is a list of operation status objects, where the status is either “failed”, “cancelled” or “success”. [{“operationid”: “opid-11ee. . . ”,“status”: “cancelled”},{“operationid”: “opid-9876”, “status”: ”failed”},{“operationid”: “opid0e0e”,“status”:”success”,“execution_time”:”25”,“result”: {“txid”:”af3887654. . . ”,. . . }},] Examples: zcashcli z_getoperationresult ‘[“opid-8120fa20-5ee7-4587-957b-f2579c2d882b”]’ zcash-cli z_getoperationresult z_getoperationstatus | [operationids] | Return OperationStatus JSON objects for all operations the node is currently aware of. Operationids is an optional array to filter which operations you want to receive status objects for. Output is a list of operation status objects.[{“operationid”: “opid-12ee. . . ”,“status”: “queued”},{“operationid”: “opd-098a. . . ”, “status”: ”executing”},{“operationid”: “opid-9876”, “status”: ”failed”}]When the operation succeeds, the status object will also include the result.{“operationid”: “opid-0e0e”,“status”:”success”,“execution_time”:”25”,“result”: {“txid”:”af3887654. . . ”,. . . }} z_listoperationids | [state] | Return a list of operationids for all operations which the node is currently aware of. State is an optional string parameter to filter the operations you want listed by their state. Acceptable parameter values are ‘queued’, ‘executing’, ‘success’, ‘failed’, ‘cancelled’.[“opid-0e0e. . . ”, “opid-1af4. . . ”, . . .]

III. Zchain ZCash block explorer API – Introduction

In This section I want to discuss the Zchain API. The Figure Below show the Zchain API for Zcash.

Figure 1: Zchain API



In order to go ahead and look at their API we need to get their website [(https://explorer.zcha.in/api).](https://explorer.zcha.in/api) To get some information about transaction, block, etc you need just type the id of the block or the hash transaction.

IV. ZCash block explorer API – Application

In this section I will you, shows how to make a small app that displays ZCash balance using PowerShell script and the zchain block explorer API.

The first step is to get access to the balance by making the following API:

```
PS C:\Users\deeplizard\zchain-app> $myAccount = Invoke-RestMethod https://api.zcha.in/v2/mainnet/accounts/t1MTLP2CtJaMDvwtSV77JHFKnXBTM4Mqq3c
PS C:\Users\deeplizard\zchain-app>
```

Furthermore to get you balance to can just type the following command:

```
PS C:\Users\deeplizard\zchain-app> $myAccount.balance
0.01569373
PS C:\Users\deeplizard\zchain-app>
```