Unauthenticated REST Interface

The REST API can be enabled with the `-rest` option.

The interface runs on the same port as the JSON-RPC interface, by default port 8332 for mainnet, port 18332 for testnet, port 38332 for signet, and port 18443 for regtest.

REST Interface consistency guarantees

The same guarantees as for the RPC Interface apply.

Limitations

There is a known issue in the REST interface that can cause a node to crash if too many http connections are being opened at the same time because the system runs out of available file descriptors. To prevent this from happening you might want to increase the number of maximum allowed file descriptors in your system and try to prevent opening too many connections to your rest interface at the same time if this is under your control. It is hard to give general advice since this depends on your system but if you make several hundred requests at once you are definitely at risk of encountering this issue.

Supported API

Transactions `GET /rest/tx/<TX-HASH>.<bin|hex|json>`

Given a transaction hash: returns a transaction in binary, hex-encoded binary, or JSON formats.

By default, this endpoint will only search the mempool. To query for a confirmed transaction, enable the transaction index via "txindex=1" command line / configuration option.

Blocks `GET /rest/block/<BLOCK-HASH>.<bin|hex|json>` `GET /rest/block/notxdetails/<BLOCK-HASH>`

Given a block hash: returns a block, in binary, hex-encoded binary or JSON formats. Responds with 404 if the block doesn't exist.

The HTTP request and response are both handled entirely in-memory.

With the /notxdetails/ option JSON response will only contain the transaction hash instead of the complete transaction details. The option only affects the JSON response.

Blockheaders `GET /rest/headers/<BLOCK-HASH>.<bin|hex|json>?count=<COUNT=5>`

Given a block hash: returns amount of blockheaders in upward direction. Returns empty if the block doesn't exist or it isn't in the active chain.

Deprecated (but not removed) since 24.0: `GET /rest/headers/<COUNT>/<BLOCK-HASH>.<bin|hex|json>`

Blockfilter Headers `GET /rest/blockfilterheaders/<FILTERTYPE>/<BLOCK-HASH>.<bin|hex|json>?co`

Given a block hash: returns amount of blockfilter headers in upward direction for the filter type . Returns empty if the block doesn't exist or it isn't in the active chain.

Deprecated (but not removed) since 24.0: `GET /rest/blockfilterheaders/<FILTERTYPE>/<COUNT>/<BLOCK-HA`

Blockfilters `GET /rest/blockfilter/<FILTERTYPE>/<BLOCK-HASH>.<bin|hex|json>`

Given a block hash: returns the block filter of the given block of type . Responds with 404 if the block doesn't exist.

Blockhash by height `GET /rest/blockhashbyheight/<HEIGHT>.<bin|hex|json>`

Given a height: returns hash of block in best-block-chain at height provided.

Chaininfos `GET /rest/chaininfo.json`

Returns various state info regarding block chain processing. Only supports JSON as output format. * chain : (string) current network name (main, test, signet, regtest) * blocks : (numeric) the current number of blocks processed in the server * headers : (numeric) the current number of headers we have validated * bestblockhash : (string) the hash of the currently best block * difficulty : (numeric) the current difficulty * mediantime : (numeric) the median time of the 11 blocks before the most recent block on the blockchain * verificationprogress : (numeric) estimate of verification progress [0..1] * chainwork : (string) total amount of work in active chain, in hexadecimal * pruned : (boolean) if the blocks are subject to pruning * pruneheight : (numeric) highest block available * softforks : (array) status of softforks in progress

Query UTXO set `GET /rest/getutxos/<checkmempool>/<txid>-<n>/<txid>-<n>/.../<txid>-<n>.<bi`

The getutxo command allows querying of the UTXO set given a set of outpoints. See BIP64 for input and output serialisation: https://github.com/bitcoin/bips/blob/master/bip-0064.mediawiki

Example:

```
$ curl localhost:18332/rest/getutxos/checkmempool/b2cdfd7b89def827ff8af7cd9bff7627ff72e5e8b0
{
   "chainHeight" : 325347,
   "chaintipHash" : "00000000fb01a7f3745a717f8caebee056c484e6e0bfe4a9591c235bb70506fb",
   "bitmap": "1",
   "utxos" : [
      {
         "height" : 2147483647,
         "value" : 8.8687,
         "scriptPubKey" : {
            "asm" : "OP_DUP OP_HASH160 1c7cebb529b86a04c683dfa87be49de35bcf589e OP_EQUALVERI
            "desc" : "addr(mi7as51dvLJsizWnTMurtRmrP8hG2m1XvD)#gj9tznmy"
            "hex" : "76a9141c7cebb529b86a04c683dfa87be49de35bcf589e88ac",
            "type" : "pubkeyhash",
            "address" : "mi7as51dvLJsizWnTMurtRmrP8hG2m1XvD"
         }
      }
   ]
}
```

Memory pool `GET /rest/mempool/info.json`

Returns various information about the TX mempool. Only supports JSON as output format. Refer to the `getmempoolinfo` RPC for documentation of the fields.

`GET /rest/mempool/contents.json`

Returns transactions in the TX mempool. Only supports JSON as output format.

Risks

Running a web browser on the same node with a REST enabled bitcoind can be a risk. Accessing prepared XSS websites could read out tx/block data of your node by placing links like `<script src="http://127.0.0.1:8332/rest/tx/1234567890.json">` which might break the nodes privacy.