```
BIP: 22
Layer: API/RPC
Title: getblocktemplate - Fundamentals
Author: Luke Dashjr <luke+bip22@dashjr.org>
Comments-Summary: No comments yet.
Comments-URI: https://github.com/bitcoin/bips/wiki/Comments:BIP-0022
Status: Final
Type: Standards Track
Created: 2012-02-28
License: BSD-2-Clause
```

Abstract

This BIP describes a new JSON-RPC method for "smart" Bitcoin miners and proxies. Instead of sending a simple block header for hashing, the entire block structure is sent, and left to the miner to (optionally) customize and assemble.

Copyright

Specification

Block Template Request

A JSON-RPC method is defined, called "getblocktemplate". It accepts exactly one argument, which MUST be an Object of request parameters. If the request parameters include a "mode" key, that is used to explicitly select between the default "template" request or a "proposal".

Block template creation can be influenced by various parameters:

template request
Key
capabilities
mode

getblocktemplate MUST return a JSON Object containing the following keys:

template
Key
bits
curtime
height
previousblockhash

template
sigoplimit
sizelimit
transactions
version
coinbaseaux
coinbasetxn
coinbasevalue
workid

Transactions Object Format The Objects listed in the response's "transactions" key contains these keys:

template "transactions" element
Key
data
depends
fee
hash
required
sigops

Only the "data" key is required, but servers should provide the others if they are known.

Block Submission

A JSON-RPC method is defined, called "submitblock", to submit potential blocks (or shares). It accepts two arguments: the first is always a String of the hex-encoded block data to submit; the second is an Object of parameters, and is optional if parameters are not needed.

submitblock parameters (2nd argument)
Key
workid

This method MUST return either null (when a share is accepted), a String describing briefly the reason the share was rejected, or an Object of these with a key for each merged-mining chain the share was submitted to.

Optional: Long Polling

template request
Key
capabilities
longpollid

template
Key
longpollid
longpolluri
submitold

If the server supports long polling, it MUST include a "longpollid" key in block templates, and it MUST be unique for each event: any given "longpollid" should check for only one condition and not be reused. For example, a server which has a long poll wakeup only for new blocks might use the previous block hash. However, clients should not assume the "longpollid" has any specific meaning. It MAY supply the "longpolluri" key with a relative or absolute URI, which MAY specify a completely different resource than the original connection, including port number. If "longpolluri" is provided by the server, clients MUST only attempt to use that URI for longpoll requests.

Clients MAY start a longpoll request with a standard JSON-RPC request (in the case of HTTP transport, POST with data) and same authorization, setting the "longpollid" parameter in the request to the value provided by the server.

This request SHOULD NOT be processed nor answered by the server until it wishes to replace the current block data as identified by the "longpollid". Clients SHOULD make this request with a very long request timeout and MUST accept servers sending a partial response in advance (such as HTTP headers with "chunked" Transfer-Encoding), and only delaying the completion of the final JSON response until processing.

Upon receiving a completed response:

- Only if "submitold" is provided and false, the client MAY discard the results of past operations and MUST begin working on the new work immediately.
- The client SHOULD begin working on the new work received as soon as possible, if not immediately.
- The client SHOULD make a new request to the same long polling URI.

If a client receives an incomplete or invalid response, it SHOULD retry the request with an exponential backoff. Clients MAY implement this backoff with limitations (such as maximum backoff time) or any algorithm as deemed suitable.

It is, however, forbidden to simply retry immediately with no delay after more than one failure. In the case of a "Forbidden" response (for example, HTTP 403), a client SHOULD NOT attempt to retry without user intervention.

Optional: Template Tweaking

template request
Key
sigoplimit
sizelimit
maxversion

For "sigoplimit" and "sizelimit", negative values and zero are offset from the server-determined block maximum. If a Boolean is provided and true, the default limit is used; if false, the server is instructed not to use any limits on returned template. Servers SHOULD respect these desired maximums, but are NOT required to: clients SHOULD check that the returned template satisfies their requirements appropriately.

Appendix: Example Rejection Reasons

Possible reasons a share may be rejected include, but are not limited to:

share rejection reasons
Reason
bad-cb-flag
bad-cb-length
bad-cb-prefix
bad-diffbits
bad-prevblk
bad-txnmrklroot
bad-txns
bad-version
duplicate
high-hash
rejected
stale-prevblk
stale-work
time-invalid
time-too-new
time-too-old
unknown-user
unknown-work

Motivation

bitcoind's JSON-RPC server can no longer support the load of generating the work required to productively mine Bitcoin, and external software specializing in work generation has become necessary. At the same time, new independent node implementations are maturing to the point where they will also be able to support miners.

A common standard for communicating block construction details is necessary to ensure compatibility between the full nodes and work generation software.

Rationale

Why not just deal with transactions as hashes (txids)?

- Servers might not have access to the transaction database, or miners may wish to include transactions not broadcast to the network as a whole.
- Miners may opt not to do full transaction verification, and may not have access to the transaction database on their end.

What is the purpose of "workid"?

- If servers allow all mutations, it may be hard to identify which job it is based on. While it may be possible to verify the submission by its content, it is much easier to compare it to the job issued. It is very easy for the miner to keep track of this. Therefore, using a "workid" is a very cheap solution to enable more mutations.

Why should "sigops" be provided for transactions?

- Due to the BIP 0016 changes regarding rules on block sigops, it is impossible to count sigops from the transactions themselves (the sigops in the scriptCheck must also be included in the count).

Reference Implementation

- Eloipool (server)
- libblkmaker (client)
- bitcoind (minimal server)

See Also

- BIP 23: getblocktemplate - Pooled Mining