```
BIP: 23
Layer: API/RPC
Title: getblocktemplate - Pooled Mining
Author: Luke Dashjr <luke+bip22@dashjr.org>
Comments-Summary: No comments yet.
Comments-URI: https://github.com/bitcoin/bips/wiki/Comments:BIP-0023
Status: Final
Type: Standards Track
Created: 2012-02-28
License: BSD-2-Clause
```

Abstract

This BIP describes extensions to the getblocktemplate JSON-RPC call to enhance pooled mining.

Copyright

Specification

Note that all sections of this specification are optional extensions on top of BIP 22.

Summary Support Levels

Something can be said to have BIP 23 Level 1 support if it implements at least:

- RFC 1945
- JSON-RPC 1.0
- BIP 22 (non-optional sections)
- BIP 22 Long Polling
- BIP 23 Basic Pool Extensions
- BIP 23 Mutation "coinbase/append"
- BIP 23 Submission Abbreviation "submit/coinbase"
- BIP 23 Mutation "time/increment" (only required for servers)

It can be said to have BIP 23 Level 2 support if it also implements:

- BIP 23 Mutation "transactions/add"
- BIP 23 Block Proposals

Basic Pool Extensions

template request
Key

template request
target

template
Key
expires
target

Block Proposal

Servers may indicate support for proposing blocks by including a capability string in their original template:

template
Key
capabilities
reject-reason

If supported, a miner MAY propose a block to the server for general validation at any point before the job expires. This is accomplished by calling getblocktemplate with two keys:

getblocktemplate parameters
Key
data
mode
workid

The block data MUST be validated and checked against the server's usual acceptance rules (excluding the check for a valid proof-of-work). If it is found to be in violation of any of these rules, the server MUST return one of the following:

- Null if it is acceptable as-is, with the same workid (if any) as provided. Note that this SHOULD NOT invalidate the old template's claim to the same workid.
- A String giving the reason for the rejection (see example rejection reasons).
- A "delta" block template (with changes needed); in this case, any missing keys are assumed to default to those in the proposed block or, if not applicable, the original block template it was based on. This template MAY also include a "reject-reason" key with a String of the reason for rejection.

It is RECOMMENDED that servers which merely need to track the proposed block for later share/* submissions, return a simple Object of the form:

```
{"workid":"new workid"}
```

Clients SHOULD assume their proposed block will remain valid if the only changes they make are to the portion of the coinbase scriptSig they themselves provided (if any) and the time header. Servers SHOULD NOT break this assumption without good cause.

Mutations

template request
Key
nonces

template
Key
maxtime
maxtimeoff
mintime
mintimeoff
mutable
noncerange

If the block template contains a "mutable" key, it is a list of these to signify modifications the miner is allowed to make:

mutations
Value
coinbase/append
coinbase
generation
time/increment
time/decrement
time
transactions/add (or "transactions")
prevblock
version/force
version/reduce

Submission Abbreviation

template
Key
fulltarget
mutable

If the block template contains a "mutable" key, it is a list of these to signify modifications the miner is allowed to make:

abbreviation mutations
Value
submit/hash
submit/coinbase
submit/truncate
share/coinbase
share/merkle
share/truncate

Format of Data for Merkle-Only Shares The format used for submitting shares with the "share/merkle" mutation shall be the 80-byte block header, the total number of transactions encoded in Bitcoin variable length number format, the coinbase transaction, and then finally the little-endian SHA256 hashes of each link in the merkle chain connecting it to the merkle root.

Logical Services

template request
Key
capabilities

template
Key
serverlist

If the "serverlist" parameter is provided, clients MAY choose to intelligently treat the server as part of a larger single logical service.

Each host Object in the Array is comprised of the following fields:

serverlist element
Key
uri
avoid
priority
sticky
update
weight

When choosing which actual server to get the next job from, URIs MUST be tried in order of their "priority" key, lowest Number first. Where the priority of URIs is the same, they should be chosen from in random order, weighed by their "weight" key. Work proposals and submissions MUST be made to the same server that issued the job. Clients MAY attempt to submit to other servers if, and only if, the original server cannot be reached. If cross-server share submissions are desired, services SHOULD instead use the equivalent domain name system (DNS) features (RFCs 1794 and 2782).

Updates to the Logical Service server list may only be made by the original server, or servers listed with the "update" key missing or true. Clients MAY choose to advertise serverlist capability to servers with a false "update" key, but if so, MUST treat the server list provided as a subset of the current one, only considered in the context of this server. At least one server with "update" privilege MUST be attempted at least once daily.

If the "sticky" key is provided, then when that server is used, it should be used consistently for at least that many seconds, if possible.

A permanent change in server URI MAY be indicated with a simple "serverlist" parameter:

```
"serverlist":[{"uri": "http://newserver"}]
```

A temporary delegation to another server for 5 minutes MAY be indicated likewise:

```
"serverlist":[{"uri": "", avoid: 300}, {"uri": "http://newserver", "update": false}]
```

Motivation

There is reasonable concerns about mining currently being too centralized on pools, and the amount of control these pools hold. By exposing the details of the block proposals to the miners, they are enabled to audit and possibly modify the block before hashing it.

To encourage widespread adoption, this BIP should be a complete superset of the existing centralized getwork protocol, so pools are not required to make substantial changes to adopt it.

Rationale

Why allow servers to restrict the complete coinbase and nonce range?

- This is necessary to provide a complete superset of JSON-RPC getwork functionality, so that pools may opt to enable auditing without significantly changing or increasing the complexity of their share validation or mining policies.
- Since noncerange is optional (both for getwork and this BIP), neither clients nor servers are required to support it.

Why specify "time/*" mutations at all?

- In most cases, these are implied by the mintime/mintimecur/maxtime/maxtimecur keys, but there may be cases that there are no applicable minimums/maximums.

What is the purpose of the "prevblock" mutation?

- There are often cases where a miner has processed a new block before the server. If the server allows "prevblock" mutation, the miner may begin mining on the new block immediately, without waiting for a new template.

Why must both "mintime"/"maxtime" and "mintimeoff"/"maxtimeoff" keys be defined?

- In some cases, the limits may be unrelated to the current time (such as the Bitcoin network itself; the minimum is always a fixed median time)
- In other cases, the limits may be bounded by other rules (many pools limit the time header to within 5 minutes of when the share is submitted to them).

Is "target" really needed?

- Some pools work with lower targets, and should not be expected to waste bandwidth ignoring shares that don't meet it.
- Required to be a proper superset of getwork.
- As mining hashrates grow, some miners may need the ability to request a lower target from their pools to be able to manage their bandwidth use.

What is the purpose of the "hash" transaction list format?

- Non-mining tools may wish to simply get a list of memory pool transactions.
- Humans may wish to view their current memory pool.

Reference Implementation

- libblkmaker
- Eloipool
- bitcoind

See Also

- BIP 22: getblocktemplate - Fundamentals