```
BIP: 300
Layer: Consensus (soft fork)
Title: Hashrate Escrows (Consensus layer)
Author: Paul Sztorc <truthcoin@gmail.com>
        CryptAxe <cryptaxe@gmail.com>
Comments-Summary: No comments yet.
Comments-URI: https://github.com/bitcoin/bips/wiki/Comments:BIP-0300
Status: Draft
Type: Standards Track
Created: 2017-08-14
License: BSD-2-Clause
Post-History: https://lists.linuxfoundation.org/pipermail/bitcoin-dev/2017-May/014364.html
```

Abstract

In Bip300, txns are not signed via cryptographic key. Instead, they are "signed" by hashpower, over time. Like a big multisig, 13150-of-26300, where each block is a new "signature".

Bip300 emphasizes slow, transparent, auditable transactions which are easy for honest users to get right and very hard for dishonest users to abuse. The chief design goal for Bip300 is *partitioning* -- users may safely ignore Bip300 txns if they want to (or Bip300 entirely).

See this site for more information.

Motivation

As Reid Hoffman wrote in 2014: "Sidechains allow developers to add features and functionality to the Bitcoin universe without actually modifying the Bitcoin Core code...Consequently, innovation can occur faster, in more flexible and distributed ways, without losing the synergies of a common platform with a single currency."

Today, coins such as Namecoin, Monero, ZCash, and Sia, offer features that Bitcoiners cannot access -- not without selling their BTC to invest in a rival monetary unit. According to coinmarketcap.com, there is now more value *outside* the BTC protocol than within it. According to cryptofees.info, 15x more txn fees are paid outside the BTC protocol, than within it.

Software improvements to Bitcoin rely on developer consensus -- BTC will pass on a good idea if it is even slightly controversial. Development is slow: we are now averaging one major feature every 5 years.

Sidechains allow for competitive "benevolent dictators" to create a new sidechain at any time. These dictators are accountable only to their users, and (crucially) they are protected from rival dictators. Users can move their BTC among these different pieces of software, as *they* see fit.

BTC can copy every useful technology, as soon as it is invented; scamcoins lose their justification and become obsolete; and the community can be pro-creativity, knowing that Layer1 is protected from harmful changes.

Specification

Overview

Bip300 allows for six new blockchain messages (these have consensus significance):

- M1. "Propose New Sidechain"
- M2. "ACK Proposal"
- M3. "Propose Bundle"
- M4. "ACK Bundle"
- M5. Deposit -- a transfer of BTC from-main-to-side
- M6. Withdrawal -- a transfer of BTC from-side-to-main

Nodes organize those messages into two caches:

- D1. "The Sidechain List", which tracks the 256 Hashrate Escrows (Escrows are slots that a sidechain can live in).
- D2. "The Withdrawal List", which tracks the withdrawal-Bundles (coins leaving a Sidechain).

D1 (The Sidechain List) D1 is a list of active sidechains. D1 is updated via M1 and M2.

Field No.	Label	Type	Description / Purpose
1	Escrow Number	uint8_t	The escrow's ID number. Used to uniquely refer to each sid
2	Version	int32_t	Version number.
3	Sidechain Name	string	A human-readable name of the sidechain.
4	Sidechain Description	string	A human-readable name description of the sidechain.
5	Hash1 - tarball hash	uint256	Intended as the sha256 hash of the tar.gz of the canonical s
6	Hash2 - git commit hash	uint160	Intended as the git commit hash of the canonical sidechain
7	Active	bool	Does this sidechain slot contain an active sidechain?
8	Activation Status	int , int	The age of the proposal (in blocks); and the number of "fail
9	"CTIP" -- "TxID"	uint256	A UTXO that holds the sidechain's money. (Part 1 of 2).
10	"CTIP" -- "vout"	int32_t	A UTXO that holds the sidechain's money. (Part 2 of 2).

D2 (The Withdrawal List) D2 lists withdrawal-attempts. If these attempts succeed, they will pay coins "from" a Bip300-locked UTXO, to new UTXOs controlled by the withdrawing-user. Each attempt pays out many users, so we call these withdrawal-attempts "Bundles".

D2 is driven by M3, M4, M5, and M6. Those messages enforce the following principles:

1. The Bundles have a canonical order (first come first serve).
2. From one block to the next, every "Blocks Remaining" field decreases by 1.
3. When "Blocks Remaining" reaches zero the Bundle is removed.
4. From one block to the next, the value in "ACKs" may either increase or decrease, by a maximum of 1 (see M4).
5. If a Bundle's "ACKs" reach 13150 or greater, it "succeeds" and its corresponding M6 message can be included in a block.
6. If the M6 of a Bundle is paid out, it is also removed.
7. If a Bundle cannot possibly succeed (13150 - "ACKs" > "Blocks Remaining"), it is removed immediately.

Field No.	Label	Type	Description / Purpose
1	Sidechain Number	uint8_t	Links the withdrawal-request to a specific hashrate escrow.
2	Bundle Hash	uint256	A withdrawal attempt. Specifically, it is a "blinded transaction
3	Work Score (ACKs)	uint16_t	How many miner upvotes a withdrawal has. Starts at 0. Fastes
4	Blocks Remaining	uint16_t	How long this bundle has left to live (measured in blocks). Sta

D1, with all 256 slots active, reaches a maximum size of: 256 * (1 (map index) + 36 (outpoint) + 8 (amount)) = 11,520 bytes.

D2, under normal conditions, would reach a size of: (38 bytes per withdrawal * 256 sidechains) = 9,728 bytes.

It is possible to spam D2. A miner can add the max M3s (256) every block, forever. This costs 9,728 on-chain bytes per block, an opportunity cost of about 43 txns. It results in no benefit to the miner whatsoever. D2 will eventually hit a ceiling at 124.5568 MB. (By comparison, the Bitcoin UTXO set is about 7,000 MB.) When the attacker stops, D2 will eventually shrink back down to 9,728 bytes.

The Six New Bip300 Messages

First, how are new sidechains created?

They are first proposed (with M1), and later acked (with M2). This process resembles Bip9 soft fork activation.

M1 -- Propose Sidechain M1 is a coinbase OP Return output containing the following:

```
   1-byte - OP_RETURN (0x6a)
4-byte - Message header (0xD5E0C4AF)
N-byte - The serialization of the sidechain.
1-byte nSidechain
4-byte nVersion
x-byte title
```

```
x-byte description
32-byte hashID1
20-byte hashID2
```

M1 is invalid if:

- It would add a duplicate entry to D1.
- There is already an M1 in this block.
- The sidechain serialization does not parse.

Otherwise:

- A new entry is added to D1, whose initial Activation Status is (age=0, fails=0).

M2 -- ACK Sidechain Proposal M2 is a coinbase OP Return output containing the following:

```
   1-byte - OP_RETURN (0x6a)
4-byte - Message header (0xD6E1C5BF)
32-byte - the sha256D hash of sidechain's serialization
```

M2 is ignored if it doesn't parse, or if it is for a sidechain that doesn't exist.

M2 is invalid if:

- An M2 is already in this block.
- It tries to ACK two different M1s for the same slot.

Otherwise:

- The sidechain is "ACK"ed and does NOT get a "fail" for this block. (As it otherwise would.)

A sidechain fails to activate if:

- If the slot is unused: during the next 2016 blocks, it accumulates 201 fails. (Ie, 90% threshold).
- If the slot is in use: during the next 26,300 blocks, it accumulates 13,150 fails. (Ie, 50% threshold).

(Thus we can overwrite a used sidechain slot. Bip300 sidechains are already vulnerable to one catastrophe per 13150 blocks (the invalid withdrawal) so this slot-overwrite option does not change the security assumptions.)

Otherwise, the sidechain activates (Active is set to TRUE).

In the block in which the sidechain activates, the coinbase MUST include at least one 0-valued OP_TRUE. This output becomes the initial CTIP for the sidechain.

Notes on Withdrawing Coins Bip300 withdrawals ("M6") are very significant.

For an M6 to be valid, it must be first "prepped" by one M3 and then 13,150+ M4s. M3 and M4 are about "Bundles".

What are Bundles? Sidechain withdrawals take the form of "Bundles" -- named because they "bundle up" many individual withdrawal-requests into a single rare layer1 transaction.

Sidechain full nodes aggregate the withdrawal-requests into a big set. The sidechain calculates what M6 would have to look like, to pay all of these withdrawal-requests out. Finally, the sidechain calculates what the hash of this M6 would be. This 32-byte hash identifies the Bundle.

This 32-byte hash is what miners will be slowly ACKing over 3-6 months, not the M6 itself (nor any sidechain data, of course).

A bundle either pays all its withdrawals out (via M6), or else it fails (and pays nothing out).

===== Bundle Hash = Blinded TxID of M6 =====

The Bundle hash is static as it is being ACKed. Unfortunately, the M6 TxID will be constantly changing -- as users deposit to the sidechain, the input to M6 will change.

To solve this problem, we do something conceptually similar to AnyPrevOut (BIP 118). We define a "blinded TxID" as a way of hashing a txn, in which some bytes are first overwritten with zeros. These are: the first input and the first output. Via the former, a sidechain can accept deposits, even if we are acking a TxID that spends from it later. Via the latter, we can force all of the non-withdrawn coins to be returned to the sidechain (even if we don't yet know how many coins this will be).

M3 -- Propose Bundle M3 is a coinbase OP Return output containing the following:

```
   1-byte - OP_RETURN (0x6a)
4-byte - Commitment header (0xD45AA943)
32-byte - The Bundle hash, to populate a new D2 entry
1-byte - nSidechain (the slot number)
```

M3 is ignored if it does not parse, or if it is for a sidechain that doesn't exist.

M3 is invalid if:

- This block already has an M3 for that nSidechain.
- A bundle with this hash is already in D2.
- A bundle with this hash already paid out.
- A bundle with this hash was rejected in the past.

Otherwise: M3 adds an entry to D2, with initial ACK score = 1 and initial Blocks Remaining = 26,299. (Merely being added to D2, does count as your first upvote.)

Once a Bundle is in D2, how can we give it enough ACKs to make it valid?

M4 -- ACK Bundle(s) M4 is a coinbase OP Return output containing the following:

```
   1-byte - OP_RETURN (0x6a)
4-byte - Commitment header (0xD77D1776)
1-byte - Version
n-byte - The "upvote vector" -- describes which bundle-choice is "upvoted", for each sidecha
```

The upvote vector will code "abstain" as 0xFF (or 0xFFFF); it will code "alarm" as 0xFE (or 0xFFFE). Otherwise it simply indicates which withdrawal-bundle in the list, is the one to be "upvoted".

For example: if there are two sidechains, and we wish to upvote the 7th bundle on sidechain #1 plus the 4th bundle on sidechain #2, then the upvote vector would be { 07, 04 }. And M4 would be [0x6A,D77D1776,00,0006,0003].

The version number allows us to shrink the upvote vector in many cases. Version 0x00 requires a full two bytes per sidechain, but it always works. Version 0x01 uses half that (one byte per sidechain), and it works while all sidechains have fewer than 255 disputed withdrawals (ie, 99.99%+ of the time). Version 0x02 uses zero bytes (ie, 6 bytes for the whole M4) and sets this block's M4 equal to the previous block's M4. Version 0x03 upvotes only those withdrawals that are leading their rivals by at least 50 votes.

If a sidechain has no pending bundles, then it is skipped over when M4 is created and parsed.

For example, an upvote vector of { 2 , N/A, 1 } would be represented as [0x6A,D77D1776,01,00,00]. It means: "upvote the first bundle in sidechain #1; and the first bundle in sidechain #3" (iff sidechains #2 has no bundles proposed).

An upvote vector of { N/A, N/A, 4 } would be [0x6A,D77D1776,01,03].

The M4 message will be invalid (and invalidate the block), if:

- It tries to upvote a Bundle that doesn't exist. (For example, trying to upvote the 7th bundle on sidechain #2, when sidechain #2 has only three bundles.)
- There are no Bundles at all, from any sidechain.

If M4 is NOT present in a block, then it is treated as "abstain".

If M4 is present and valid: each withdrawal-bundle that is ACKed, will gain one upvote.

Important: Within a sidechain-group, upvoting one Bundle ("+1") automatically downvotes ("-1") all other Bundles in that group. However, the minimum ACK-counter is zero. While only one Bundle can be upvoted at once; the whole group can all be unchanged at once ("abstain"), and they can all be downvoted at once ("alarm").

For example:

SC#	Bundle Hash	ACKs	Blocks Remaining
1	h1	45	22,109
1	h2	12	22,008
2	h3	13	22,999
2	h4	8	23,550
2	h5	2	22,560

...in block 900,000 could become...

SC#	Bundle Hash	ACKs	Blocks Remaining
1	h1	46	22,108
1	h2	11	22,007
2	h3	12	22,998
2	h4	9	23,549
2	h5	1	22,559

...if M4 were [0x6A,D77D1776,00,0000,0001].

Finally, we describe Deposits and Withdrawals.

M5 -- Deposit BTC to Sidechain Each sidechain stores all its BTC in one UTXO, called the "CTIP".

By definition, an M5 is a transaction which spends the CTIP and **increases** the quantity of coins. An M6 is a transaction which spends the CTIP and **decreases** the quantity of coins in the CTIP. See here.

Every time a deposit/withdrawal is made, the old CTIP is spent and a new one is created. (Deposits/Withdrawals never cause UTXO bloat.) At all times, the CTIP of each sidechain is cached in D1 (above).

Every M5 is valid, as long as:

- It has at least one OP_TRUE output -- this becomes the new CTIP.
- The new CTIP has **more** coins in it, than before.

M6 -- Withdraw BTC from a Sidechain We come, finally, to the critical matter: where users can take their money *out* of the sidechain.

M6 is invalid if:

- The blinded hash of M6 does NOT match one of the approved Bundle-hashes. (In other words: M6 must first be approved by 13,150 upvotes.)
- The first output of M6 is NOT an OP_TRUE. (This OP_TRUE becomes the new CTIP. In other words: all non-withdrawn coins are paid back to the sidechain.)
- The second output is NOT an OP_RETURN script of exactly 10 bytes, of which 8 bytes are a serialized Bitcoin amount.
- The txn fee of M6 is NOT exactly equal to the amount of the previous bullet point.

Else, M6 is valid.

(The point of the latter two bullet points, is to allow the bundle hash to cover the L1 transaction fee.)

Backward compatibility

As a soft fork, older software will continue to operate without modification. Non-upgraded nodes will see a number of phenomena that they don't understand -- coinbase txns with non-txn data, value accumulating in anyone-can-spend UTXOs for months at a time, and then random amounts leaving these UTXOs in single, infrequent bursts. However, these phenomena don't affect them, or the validity of the money that they receive.

(As a nice bonus, note that the sidechains themselves inherit a resistance to hard forks. The only way to guarantee that all different sidechain-nodes will always report the same Bundle, is to upgrade sidechains via soft forks of themselves.)

Deployment

This BIP will be deployed via UASF-style block height activation. Block height TBD.

Reference Implementation

See: https://github.com/drivechain-project/mainchain

Also, for interest, see an example sidechain here: https://github.com/drivechain-project/sidechains/tree/testchain

References

https://github.com/drivechain-project/mainchain https://github.com/drivechain-project/sidechains/tree/testchain See http://www.drivechain.info/

literature/index.html

Credits

Thanks to everyone who contributed to the discussion, especially: Luke Dashjr, ZmnSCPxj, Adam Back, Peter Todd, Dan Anderson, Sergio Demian Lerner, Chris Stewart, Matt Corallo, Sjors Provoost, Tier Nolan, Erik Aronesty, Jason Dreyzehner, Joe Miyamoto, Ben Goldhaber.

Copyright