```
BIP: 91
Layer: Consensus (soft fork)
Title: Reduced threshold Segwit MASF
Author: James Hilliard <james.hilliard1@gmail.com>
Comments-Summary: No comments yet.
Comments-URI: https://github.com/bitcoin/bips/wiki/Comments:BIP-0091
Status: Final
Type: Standards Track
Created: 2017-05-22
License: BSD-3-Clause
         CC0-1.0
```

Abstract

This document specifies a method to activate the existing BIP9 segwit deployment with a majority hashpower less than 95%.

Definitions

"existing segwit deployment" refer to the BIP9 "segwit" deployment using bit 1, between November 15th 2016 and November 15th 2017 to activate BIP141, BIP143 and BIP147.

Motivation

Segwit increases the blocksize, fixes transaction malleability, and makes scripting easier to upgrade as well as bringing many other benefits.

This BIP provides a way for a simple majority of miners to coordinate activation of the existing segwit deployment with less than 95% hashpower. For a number of reasons a complete redeployment of segwit is difficult to do until the existing deployment expires. This is due to 0.13.1+ having many segwit related features active already, including all the P2P components, the new network service flag, the witness-tx and block messages, compact blocks v2 and preferential peering. A redeployment of segwit will need to redefine all these things and doing so before expiry would greatly complicate testing.

Specification

While this BIP is active, all blocks must set the nVersion header top 3 bits to 001 together with bit field (1«1) (according to the existing segwit deployment). Blocks that do not signal as required will be rejected.

Deployment

This BIP will be deployed by a "version bits" with an 80%(this can be adjusted if desired) 269 block activation threshold and 336 block confirmation window

BIP9 with the name "segsignal" and using bit 4.

This BIP will have a start time of midnight June 1st, 2017 (epoch time 1496275200) and timeout on midnight November 15th 2017 (epoch time 1510704000). This BIP will cease to be active when segwit (BIP141) is locked-in, active, or failed

Reference implementation

```
// Deployment of SEGSIGNAL
consensus.vDeployments[Consensus::DEPLOYMENT_SEGSIGNAL].bit = 4;
consensus.vDeployments[Consensus::DEPLOYMENT_SEGSIGNAL].nStartTime = 1496275200; // June 1st
consensus.vDeployments[Consensus::DEPLOYMENT_SEGSIGNAL].nTimeout = 1510704000; // November 1
consensus.vDeployments[Consensus::DEPLOYMENT_SEGSIGNAL].nOverrideMinerConfirmationWindow = 3
consensus.vDeployments[Consensus::DEPLOYMENT_SEGSIGNAL].nOverrideRuleChangeActivationThresho

class VersionBitsConditionChecker : public AbstractThresholdConditionChecker {
private:
    const Consensus::DeploymentPos id;

protected:
    int64_t BeginTime(const Consensus::Params& params) const { return params.vDeployments[id
    int64_t EndTime(const Consensus::Params& params) const { return params.vDeployments[id].
    int Period(const Consensus::Params& params) const {
        if (params.vDeployments[id].nOverrideMinerConfirmationWindow > 0)
            return params.vDeployments[id].nOverrideMinerConfirmationWindow;
        return params.nMinerConfirmationWindow;
    }
    int Threshold(const Consensus::Params& params) const {
        if (params.vDeployments[id].nOverrideRuleChangeActivationThreshold > 0)
            return params.vDeployments[id].nOverrideRuleChangeActivationThreshold;
        return params.nRuleChangeActivationThreshold;
    }

    bool Condition(const CBlockIndex* pindex, const Consensus::Params& params) const
    {
        return (((pindex->nVersion & VERSIONBITS_TOP_MASK) == VERSIONBITS_TOP_BITS) && (pind
    }

public:
    VersionBitsConditionChecker(Consensus::DeploymentPos id_) : id(id_) {}
    uint32_t Mask(const Consensus::Params& params) const { return ((uint32_t)1) << params.vD
};

// SEGSIGNAL mandatory segwit signalling.
if (VersionBitsState(pindex->pprev, chainparams.GetConsensus(), Consensus::DEPLOYMENT_SEGSIG
```

```
    VersionBitsState(pindex->pprev, chainparams.GetConsensus(), Consensus::DEPLOYMENT_SEGWIT
{
    bool fVersionBits = (pindex->nVersion & VERSIONBITS_TOP_MASK) == VERSIONBITS_TOP_BITS;
    bool fSegbit = (pindex->nVersion & VersionBitsMask(chainparams.GetConsensus(), Consensus
    if (!(fVersionBits && fSegbit)) {
        return state.DoS(0, error("ConnectBlock(): relayed block must signal for segwit, ple
    }
}
```

https://github.com/segsignal/bitcoin

Backwards Compatibility

This deployment is compatible with the existing "segwit" bit 1 deployment scheduled between midnight November 15th, 2016 and midnight November 15th, 2017. Miners will need to upgrade their nodes to support segsignal otherwise they may build on top of an invalid block. While this bip is active users should either upgrade to segsignal or wait for additional confirmations when accepting payments.

Rationale

Historically we have used IsSuperMajority() to activate soft forks such as BIP66 which has a mandatory signalling requirement for miners once activated, this ensures that miners are aware of new rules being enforced. This technique can be leveraged to lower the signalling threshold of a soft fork while it is in the process of being deployed in a backwards compatible way.

By orphaning non-signalling blocks during the BIP9 bit 1 "segwit" deployment, this BIP can cause the existing "segwit" deployment to activate without needing to release a new deployment.

References

- Mailing list discussion
- P2SH flag day activation
- BIP9 Version bits with timeout and delay
- BIP16 Pay to Script Hash
- BIP141 Segregated Witness (Consensus layer)
- BIP143 Transaction Signature Verification for Version 0 Witness Program
- BIP147 Dealing with dummy stack element malleability
- BIP148 Mandatory activation of segwit deployment
- BIP149 Segregated Witness (second deployment)
- Segwit benefits

Copyright