```
BIP: 131
Layer: Consensus (hard fork)
Title: "Coalescing Transaction" Specification (wildcard inputs)
Author: Chris Priest <cp368202@ohiou.edu>
Comments-Summary: No comments yet.
Comments-URI: https://github.com/bitcoin/bips/wiki/Comments:BIP-0131
Status: Rejected
Type: Standards Track
Created: 2015-11-30
License: PD
```

Abstract

This specification defines a new type of transaction that supplements (not replaces) normal "non coalescing" bitcoin transactions.

Motivation

Normal "non-coalescing" Bitcoin Transactions have one large inefficiency: When you want to spend from multiple inputs with the exact same scriptPubKey, you have to list each input separately, along with the same signature multiple times, even though the signature expresses the same information. This bloats the transaction size and makes it expensive to spend from small value inputs.

Because small value inputs are expensive to send, they remain in the UTXO pool which full nodes have to keep around. It is believed that long term increase of the UTXO set can have negative scaling consequences on the network.

If maximum blocksize is made to increase *slower* than the actual number of transactions bitcoin users are sending to the network, this problem is projected to get worse. In other words, as transaction fees increase, the minimum economical value of a spending a UTXO will increase.

Specification

Redefinition of Transaction version

First, this BIP redefines the version field on transactions. The first four bytes are defined as the version number, while the last four bytes are defined as the transaction type. Type "0000" denotes normal transactions, and "0001" defines coalescing transaction.

Examples:

version 1 transaction with normal inputs:

```
version: 10000000
```

version 2 transaction with normal inputs:

```
version: 20000000
```

version 2 transaction with coalescing inputs:

```
version: 20000001
```

Essentially the last bit in the version field is set to 1 to enable wildcard inputs for all inputs present in the transaction.

Wildcard inputs

A coalescing transaction is formulated the exact same way as a version 1 transaction with one exception: each input is treated as a "wildcard input".

A wildcard input beings the value of all inputs with the exact same scriptPubKey in a block lower or equal to the block the wildcard input is confirmed into.

Changes needed to implement

The bitcoin code needs to be modified in three places in order to handle Coalescing Transactions.

1. Full Node Coalescing validation - When a full node receives a coalescing transaction, it has to aggregate the value of all the UTXOs in the blockchain older than the input with the same scriptPubKey. If this value is greater than or equal to the amount of all outputs, then that coalescing transaction is valid and can be propagated.

2. Full Node Non-Coalescing validation - When a non-coalescing transaction comes in, the code needs to be modified to check if each input has not been spent by a coalescing transaction. If there exist any coalescing transaction in the blockchain with the same scriptPubKey found in a block *after* that input, then the UTXO has been spent and the transaction is invalid.

3. Wallet - The user facing wallet portion of the reference client should notify the user when their wallet contains many UTXOs that qualify it to benefit from a coalescing transaction. Wallets should not simply replace non-coalescing transactions with coalescing transactions in all instances.

Isn't this BIP bad because it encourage address re-use?

Address re-use comes in two forms: re-use by the *sender*, and re-use by the *receiver*.

Re-use by the sender is basically using the same address for the change output. This is generally considered bad since people looking through your transaction history can determine who you do business with. When you generate a new address for every change, your privacy is conserved as it is impossible to know which output is a recipient, and which output is the change output. This BIP has **no effect** on re-use by the sender.

On the other hand, address re-use by the *receiver* occurs under completely different circumstances. When you publish an address and have multiple people send to that address, you are engaging in address re-use from the receiver. This activity has historically been considered bad because it leads to re-using a private key. When you re-use a private key too many times, you run the risk of an attacker performing statistical analysis on the multiple signatures, which can lead to an attacker finding out your private key.

This BIP introduces a way to spend multiple inputs *without* re-using the private key. In a sense, this BIP fixes the problem that makes address re-use bad for the receiver. After this BIP becomes implemented and deployed, address re-use by the receiver will no longer be considered bad form.

Copyright