```
BIP: 74
Layer: Applications
Title: Allow zero value OP_RETURN in Payment Protocol
Author: Toby Padilla <tobypadilla@gmail.com>
Comments-Summary: Unanimously Discourage for implementation
Comments-URI: https://github.com/bitcoin/bips/wiki/Comments:BIP-0074
Status: Rejected
Type: Standards Track
Created: 2016-01-29
License: PD
```

Abstract

This BIP alters the Payment Protocol to allow for zero value OP_RETURN outputs in serialized PaymentRequests.

Motivation

The Payment Protocol (defined in BIP70) gives merchants a way to build sophisticated transactions by serializing one or more outputs in the form of a PaymentRequest. The PaymentRequest is then served over http/https to a customer's wallet where the serialized transaction can be executed.

While the Payment Protocol allows for any valid script in its outputs, it also ignores outputs with zero value. This means BIP70 implementations can encode an OP_RETURN script but must provide a greater than dust value for that output. The end result is a successful PaymentRequest transaction with an OP_RETURN but the value assigned to that output is lost forever.

This BIP allows for zero value OP_RETURN outputs in serialized PaymentRequests. The change means that OP_RETURN scripts will work as they were originally intended from within PaymentRequests without permanently destroying Bitcoin value. Zero value non-OP_RETURN scripts should continue to be ignored.

In addition to fixing the issue of destroyed value, this change opens up new use cases that were previously impossible.

While storing data on the blockchain is controversial, when used responsibly OP_RETURN provides a powerful mechanism for attaching metadata to a transaction. This BIP effectively decouples the creation of transactions containing OP_RETURN data from the execution of those transactions. The result are positive benefits for both merchants and wallets/customers.

By supporting this BIP, wallets can participate in current and future, unforeseen use cases that benefit from metadata stored in OP_RETURN. Until now OP_RETURN transactions have typically been created and submitted by custom software. If a wallet can process a PaymentRequest with OP_RETURN

data as proposed by this BIP, it will support potentially sophisticated Bitcoin applications without the wallet developer having to have prior knowledge of that application.

An example might be a merchant that adds the hash of a plain text invoice to the checkout transaction. The merchant could construct the PaymentRequest with the invoice hash in an OP_RETURN and pass it to the customer's wallet. The wallet could then submit the transaction, including the invoice hash from the PaymentRequest. The wallet will have encoded a proof of purchase to the blockchain without the wallet developer having to coordinate with the merchant software or add features beyond this BIP.

Merchants and Bitcoin application developers benefit from this BIP because they can now construct transactions that include OP_RETURN data in a keyless environment. Again, prior to this BIP, transactions that used OP_RETURN (with zero value) needed to be constructed and executed in the same software. By separating the two concerns, this BIP allows merchant software to create transactions with OP_RETURN metadata on a server without storing public or private Bitcoin keys. This greatly enhances security where OP_RETURN applications currently need access to a private key to sign transactions.

Specification

The specification for this BIP is straightforward. BIP70 should be fully implemented with the following changes:

- Outputs where the script is an OP_RETURN and the value is zero should be accepted by the wallet.

BIP70 has special handling for the case with multiple zero value outputs:

> If the sum of outputs.amount is zero, the customer will be asked how much to pay, and the bitcoin client may choose any or all of the Outputs (if there are more than one) for payment. If the sum of outputs.amount is non-zero, then the customer will be asked to pay the sum, and the payment shall be split among the Outputs with non-zero amounts (if there are more than one; Outputs with zero amounts shall be ignored).

This behavior should be retained with the exception of OP_RETURN handling. In the case of a multiple output transaction where the sum of the output values is zero, the user should be prompted for a value and that value should be distributed over any or all outputs *except* the OP_RETURN output. In the case where the sum of outputs.amount is non-zero then any OP_RETURN outputs should not be ignored but no value should be assigned to them.

Payment requests also must contain at least one payable output (i.e. no payment requests with *just* an OP_RETURN).

Rationale

As with the discussion around vanilla OP_RETURN, the practice of storing data on the blockchain is controversial. While blockchain and network bloat is an undeniable issue, the benefits that come from attaching metadata to transactions has proven to be too powerful to dismiss entirely. In the absence of OP_RETURN support the Bitcoin ecosystem has seen alternative, less elegant and more wasteful methods employed for Blockchain data storage.

As it exists today, BIP70 allows for OP_RETURN data storage at the expense of permanently destroyed Bitcoin. Even fully removing support for OP_RETURN values in the Payment Protocol would still leave the door open to suboptimal data encoding via burning a larger than dust value to an output with a false address designed to encode data.

This BIP offers all of the same benefits that come from the OP_RETURN compromise. Mainly that OP_RETURN scripts are provably unspendable and thus can be pruned from the UTXO pool. Without supporting this BIP, wallets that support BIP70 will allow for wasteful data storage.

Compatibility

Since this BIP still supports OP_RETURN statements with a greater than zero value, it should be fully backwards compatible with any existing implementations.

Copyright