```
BIP: 10
Layer: Applications
Title: Multi-Sig Transaction Distribution
Author: Alan Reiner <etotheipi@gmail.com>
Comments-Summary: No comments yet.
Comments-URI: https://github.com/bitcoin/bips/wiki/Comments:BIP-0010
Status: Withdrawn
Type: Informational
Created: 2011-10-28
```

A multi-signature transaction is one where a certain number of Bitcoins are "encumbered" with more than one recipient address. The subsequent transaction that spends these coins will require each party involved (or some subset, depending on the script), to see the proposed transaction and sign it with their private key. This necessarily requires collaboration between all parties -- to propose a distribution of encumbered funds, collect signatures from all necessary participants, and then broadcast the completed transaction.

This BIP describes a way standardize the encoding of proposal transactions, to assist with signature collection and broadcast (which includes regular, 1-of-1 transactions requiring signatures from an offline computer). The goal is to encourage a standard that guarantees interoperability of all programs that implement it.

Motivation

The enabling of multi-signature transactions in Bitcoin will introduce a great deal of extra functionality to the users of the network, but also a great deal of extra complexity. Executing a multi-signature tx will be a multi-step process, and will potentially get worse with multiple clients, each implementing this process differently. By providing an efficient, standardized technique, we can improve the chance that developers will adopt compatible protocols and not bifurcate the user-base based on client selection.

In addition to providing a general encoding scheme for transaction signing/collection, it does not require the signing device to hold any blockchain information (all information needed for verification and signing is part of the encoding). This enables the existence of very lightweight devices that can be used for signing since they do not need the blockchain -- only a minimal set of Bitcoin tools and an ECDSA module. Therefore, BIP 0010 has benefit beyond just multi-signature transactions.

Specification

This BIP proposes the following process, with terms in quotes referring to recommended terminology that should be encouraged across all implementations.

1. One party will initiate this process by creating a "Distribution Proposal", which could be abbreviated DP, or TxDP
2. The user creating the TxDP (the preparer) will create the transaction as they would like to see it spent, but with blank TxIn scripts (where the signatures scripts will eventually go).
3. The proposed transaction will be spending a set of unspent TxOuts available in the blockchain. The full transactions containing these TxOuts will be serialized and included, as well. This so that the values of the TxIns can be verified before signing (the prev-tx-hash is part of the data being signed, but the value is not). By including the full tx, the signing party can verify that the tx matches the OutPoint hash, and then verify input values, all without any access to the blockchain.
4. The TxDP will have an "DP ID" or "Unsigned ID" which is the hash of the proposed transaction with blanked scripts, in Base58. This is a specific naming convention to make sure it is not confused with the actual the transaction ID that it will have after it is broadcast (the transaction ID cannot be determined until after all signatures are collected). The final Tx ID can be referred to as its "Broadcast ID", in order to distinguish it from the pre-signed ID.
5. The TxDP will have a potentially-unordered list of sig-pubkey pairs which represent collected signatures. If you receive a TxDP missing only your signature, you can broadcast it as soon as you sign it.
6. Identical TxDP objects with different signatures can be easily combined. This allows one party to send out all the requests for signatures at once, and combine them all when they are received (instead of having to "pass it around".
7. For cases where the TxDP might be put into a file or sent via email, it should use .txdp or .btcdp suffix

Anyone adopting BIP 0010 for multi-sig transactions will use the following format (without indentation):

```
'-----BEGIN-TRANSACTION-TXDPID-------'
("_TXDIST_") (magicBytes) (base58Txid) (varIntTxSize)
   (serializedTxListInHex_Line1)
   (serializedTxListInHex_Line2)
   (serializedTxListInHex_Line3)
   ...
("_TXINPUT_") (00) (InputValue)
   ("_SIG_") (AddrBase58) (SigBytes) (SigHexPart0)
   (SigHexRemainingLines)
   ("_SIG_") (AddrBase58) (SigBytes) (SigHexPart0)
   (SigHexRemainingLines)
("_TXINPUT_") (01) (InputValue)
   ("_SIG_") (AddrBase58) (SigBytes) (SigHexPart0)
   (SigHexRemainingLines)
("_TXINPUT_") (02) (InputValue)
```

```
 '-------END-TRANSACTION-TXDPID-------'
```

The following is an example TxDP from Armory, produced while running on the test network. Its DPID is 3fX59xPj:


```
-----BEGIN-TRANSACTION-3fX59xPj-------------------------------------------------
_TXDIST_fabfb5da_3fX59xPj_00a0
010000000292807c8e70a28c687daea2998d6273d074e56fa8a55a0b10556974cf2b526e61000000
0000ffffffffe3c1ee0711611b01af3dee55b1484f0d6b65d17dce4eff0e6e06242e6cf457e10000
000000ffffffff02b0feea0b000000001976a91457996661391fa4e95bed27d7e8fe47f47cb8e428
88ac00a0acb9030000001976a914dc504e07b1107110f601fb679dd3f56cee9ff71e88ac00000000
0100000001eb626e4f73d88f415a8e8cb32b8d73eed47aa1039d0ed2f013abdc741ce6828c010000
008c493046022100b0da540e4924518f8989a9da798ca2d9e761b69a173b8cc41a3e3e3c6d77cd50
022100ecfa61730e5800533842051674 4ef680428dcfc05022dec70a851365c8575b190141042dc5
be3afa5887aee4a377032ed014361b0b9b61eb3ea6b8a8821bfe13ee4b65cd25d9630e4f227a53e8
bf637f85452c9981bcbd64ef77e22ce97b0f547c783effffffff0200d6117e030000001976a914cf
f580fd243f64f0ad7bf69faf41c0bf42d86d8988ac00205fa012000000001976a9148d573ef6984fd9
f8847d420001f7ac49b222a24988ac00000000010000001f2782db40ae147398a31cff9c7cc3423
014a073a92e463741244330cc304168f000000008c493046022100c9311b9eef0cc69219cb96838f
dd621530a80c46269a00dccc66498bc03ccf7a0221003742ee652a0a76fd28ad81aa73bb7f7a0a6a
81850af58f62d9a184d10e5eec30014104f815e8ef4cad584e04974889d7636e8933803d2e72991d
b5288c9e953c2465533905f98b7b688898c7c1f0708f2e49f0dd0abc06859ffed5144e8a1018a4e8
63ffffffff02008c8647000000001976a914d4e211215967f8e3744693bf85f47eb4ee9567fc88ac
603d4e95010000001976a914e9a6b50901c1969d2b0fd43a3ccfa3fef3291efe88ac00000000
_TXINPUT_00_150.00000000
_SIG_mzUYGfqGpyXmppYpmWJ31Y4zTxR4ZCod22_00_008c
493046022100769996 7c3ec09d072599558d2e7082fae0820206b63aa66afea124634ed11a080221
0003346f7e963e645ecae2855026dc7332eb7237012539b34cd441c3cef97fbd4d01410497d5e1a0
0e1db90e893d1f2e547e2ee83b5d6bf4ddaa3d514e6dc2d94b6bcb5a72be1fcec766b8c382502caa
9ec09fe478bad07d3f38ff47b2eb42e681c384cc
_TXINPUT_01_12.00000000
_SIG_mzvaN8JUhHLz3Gdec1zBRxs5rNaYLQnbD1_01_008c
49304602210081554f8b08a1ad8caa69e34f4794d54952dac7c5efcf2afe080985d6bd5b00770221
00dea20ca3dbae1d15ec61bec57b4b8062e7d7c47614aba032c5a32f651f471cfd014104c30936d2
456298a566aa76fefeab8a7cb7a91e8a936a11757c911b4c669f0434d12ab0936fc13986b156156f
9b389ed244bbb580112be07dbe23949a4764dffb
-------END-TRANSACTION-3fX59xPj-------------------------------------------------
```


In this transaction, there are two inputs, one of 150 BTC and the other of 12 BTC. This transaction combines 162 BTC to create two outputs, one of 160 BTC, one 1.9995 BTC, and a tx fee of 0.0005. In this TxDP, both inputs have been signed, and thus could broadcast immediately.

The style of communication is taken directly from PGP/GPG, which uses blocks of ASCII like this to communicate encrypted messages and signatures. This serialization is compact, and will be interpretted the same in all character encodings. It can be copied inline into an email, or saved in a text file. The advantage over the analogous PGP encoding is that there are some human readable elements to it, for users that wish to examine the TxDP packet manually, instead of requiring a program to parse the core elements of the TxDP.

A party receiving this TxDP can simply add their signature to the appropriate _TXINPUT_ line. If that is the last signature required, they can broadcast it themselves. Any software that implements this standard should be able to combine multiple TxDPs into a single TxDP. However, even without the programmatic support, a user could manually combine them by copying the appropriate _TXSIGS_ lines between serializations, though it is not the recommended method for combining TxDPs.

Reference Implementation

This proposal was implemented and tested in the older versions of *Armory* Bitcoin software for use in offline-wallet transaction signing (as a 1-of-1 transaction). Implementation can be found in https://github.com/etotheipi/BitcoinArmory/blob/v0.91-beta/armoryengine/Transaction.py under the class PyTxDistProposal. However, as of verion 0.92 released in July 2014, Armory no longer uses this proposal for offline wallet transaction signing and has moved on to a new format.