# **[翻译]BIP141 隔离见证交易**

## **综述**

本提案定义了一个新结构“witness”提交到区块中，与交易的merkle tree不同。该结构包含检查交易有效性所需的数据。此外，脚本和签名移动到了这个新结构。

为了确保BIP软分叉的兼容性，该结构通过coinbase交易嵌套在区块已存在的merkle root。未来的硬分叉可以给该树分配一个自己的分支。

## **动机**

整个交易的效果由引用的交易输出和新创建的交易输出决定。其他的交易数据，尤其是签名，仅仅只在验证块链的状态的时候需要，其他的时候并不需要它。

通过从交易结构中移除这些数据，提交给交易merkle tree，几个问题得到修正:

1. ****无意导致的交易延展性变的不可能**** : 因为签名数据不再是组成交易哈希数据的一部分，所以交易签名的变化将不再影响到对应交易的确定。作为一个交易延展性的解决方案，它比经典的签名编码解决方案更好(BIP62)。

* 对于所有类型的脚本，该方案阻止了无意导致的签名延展性，只要所有的交易输入被签名(使用至少一个CHECKSIG或CHECKMULTISIG操作码)。
* 在m-of-n的CHECKMULTISIG的多重签名脚本中，只有得到所有私钥所有者的同意，该交易才可能被延展(而不是BIP62中仅一个私钥的持有者).
* 防止未知的ECDSA签名延展性造成的交易延展。
* 允许没有风险的创建未确认的交易依赖链，这是对于offchain协议一个重要的功能(例如：闪电网络)

1. ****传输签名数据变得可选择**** : 签名数据只有当一个对等节点准备验证该交易有效性是才需要，对于只检查该交易是否存在，并不需要该交易的签名数据。这将减少SPV 节点证明的字节大小，提升SPV节点的隐私性，因为SPV节点可以使用相同的带宽下载更多的交易，
2. ****一些限制可以通过软分叉绕过**** : 通过转移交易的部分数据到当前协议未知的数据结构中，例如：

* 当计算区块字节大小时，witness的字节可以被忽略或减少，一定程度上增大了区块的字节。
* 硬编码常量，例如最大的push字节数或操作码个数的限制可以被移除或重新评估。
* 新的脚本系统可以不受现有脚本语义的限制被引入。例如：对于交易的签名验证，一个新的交易摘要算法被引入(BIP143)

## **规范**

### **交易ID**

一个新的数据结构：witness被定义。每个交易有两个ID。

txid的定义未变；  
[nVersion] [txins] [txouts] [nLockTime]

新的wtxid被定义：  
[nVersion] [marker] [flag] [txins] [txouts] [witness] [nLockTime]  
nVersion, txins, txouts, and nLockTime格式与传统的序列化相同。  
marker 必须是一个值为0的字节，0x00.  
flag 必须是一个非0的字节：当前，必须是0x01。  
witness是每个交易所有witness数据的序列化。每个txin关联一个witness字段。一个witness字段以var\_int开始，标识txin栈中item的数量，后面紧跟栈上的items，每个item以var\_int开始，标识长度。witness数据不是脚本。  
一个非witness程序的txin必须关联一个空的witness字段，由0x00表示。如果所有的txins都是非witness程序，则交易的wtxid等于它的txid。

### **承诺结构**

一个新的区块规则被添加，该规则要求wtxid的承诺结构。coinbase交易的承诺结构被假设为0x00...000.  
witness root hash由所有wtxid做为叶子来计算，与块头的hashMerkleRoot相同的方式。  
该承诺结构被记录在coinbase交易的scriptPubKey字段。它必须含有至少38个字节，前6个字节是：0x6a24aa21a9ed,

1-byte - OP\_RETURN (0x6a)

1-byte - Push the following 36 bytes (0x24)

4-byte - Commitment header (0xaa21a9ed)

32-byte - Commitment hash: Double-SHA256(witness root hash|witness reserved value)

39th byte onwards: Optional data with no consensus meaning

并且coinbase的交易输入必须含有一个32字节的数组，做为witness reserved value.  
如果有多个scriptPubKey匹配这个格式，则索引最大的输出被认定为承诺结构。  
如果区块的所有交易都不含witness数据，此时这个承诺结构就可有可无。

### **witness program**

一个scriptPubKey(或定义在BIP16/P2SH中的redeemScript)由一个字节的push 操作码，后紧跟一个2-40字节的数据获得一个新的特殊含义。第一个push的值叫做version byte,接下来的push字节向量叫做witness program.

有两种情况下，witness的验证逻辑被触发。每种情况都决定了witness version字节和程序的位置，以及scriptSig的格式。

1. 由一个版本字节和witness program组成的scriptPubKey触发。则scriptSig必须恰好是空或验证失败("native witness program")。
2. 被一个P2SH脚本的scriptPubKey触发，并且pushscriptSig到redeemScript正好是一个版本字节加一个witness program。这个scriptSig必须恰好是一个push到BIP16的redeemScript或验证失败("P2SH witness program")

如果版本字节是0，并且witness program是20字节

* 解释为一个 pay-to-witness-public-key-hash (P2WPKH)程序。
* 这个witness必须正好包含两个item,且每个item小于等于520字节。第一个是签名，第二个是公钥。
* 公钥的hash160必须匹配20个字节的witness program。
* 正常脚本评估后，签名和公钥通过CHECKSIG操作验证通过。验证必须在栈上chanTRUE的输出。

如果版本字节是0，并且witness program是32字节：

* 解释为一个pay-to-witness-script-hash(P2SH)程序。
* witness必须包含一个输入栈以供给脚本，然后是一个脚本的序列化(witnessScript).
* witnessScript被反序列，并在正常的脚本评估后使用剩余的witness栈继续执行(每个堆栈项小于等于520字节).
* 脚本一定不能执行失败，并在栈上产生一个TRUE。  
  如果版本字节是0，但是witness program既不是20字节，也不是32字节，这个脚本必须失败1。

如果版本字节是1到16，且witness program 或witness 堆栈没有进一步的解释，并且对于witness 堆栈没有字节限制。这些版本号保留给未来的拓展2。

### **其他的共识规则限制**

#### **Block Size**

区块当前被限制在1,000,000(1M)字节。如下这样改变限制：  
Block wight = Base size \* 3 + Total size; 3.  
Base size是不带witness数据的交易序列化后的字节大小，具体方式查看未升级节点。  
Total Size 是BIP144中描述的交易序列化方式后的字节大小，包含基础数据和witness数据。  
新的规则是：block wight <= 4,000,000(4M)

#### **Sigops**

当前每个块的操作码被限制在20,000。如下这样改变限制：  
在当前锁定脚本，签名脚本，P2SH检查脚本，操作码被计数为先前值的4倍。操作码的限制同样翻了一番 <= 80,000.  
每个P2WPKH的交易输入被记为1个操作码。另外，在P2WSHwitnessScript 中的操作码被记为与以前P2SHredeemScript中相同的个数。也就是说，CHECKSIG被记为一个操作码，CHECKMULTISIG根据参数被记为1到20个操作码。这个规则适用于标准的witness program和P2SH witness program。

### **额外的定义**

接下来的定义没有用于共识限制，但建议符合上述术语。

#### **Transaction size calculations(交易字节计算)**

Transaction weight = Base transaction size \* 3 + Total  
transaction size。(例如：相同的方法计算block weight)  
Virtual transaction size = Transaction weight / 4. (四舍五入至下一个整型)  
Base transaction size = 不带witness 数据的交易序列化后的大小。  
Total transaction size = [BIP144](https://link.jianshu.com/?t=https://github.com/bitcoin/bips/blob/master/bip-0144.mediawiki" \t "https://www.jianshu.com/p/_blank)中描述的交易序列化后的字节大小，包含基础数据和witness数据。

### **新的脚本语义**

尽管P2WPKH和P2WSH脚本看起来非常类似于预先隔离验证脚本，但还是有一些显著的不同。用户不得假设在预先隔离见证中可花费的脚本在P2WPKH或P2WSH也可以花费。再生产网络大规模部署之前，开发者应该在测试网络上使用默认的中继策略测试脚本，并且在BIP141在主网络被激活后，使用较少的金额去测试。

共识级别的主要不同在[BIP143](https://link.jianshu.com/?t=https://github.com/bitcoin/bips/blob/master/bip-0143.mediawiki%23Restrictions_on_public_key_type" \t "https://www.jianshu.com/p/_blank)中描述，在0号版本的witness program 为签名验证定义了一个新的交易摘要算法。

在参考实现版本0.13.1中，三个中继策略和挖矿政策也被包含在第一版的隔离见证中。基于这些策略的软分叉很可能在不久的将来提出。为了在软分叉中造成避免无限期的延迟交易确认或永久性的资金丢失，用户必须仔细查看新的脚本语义。

1. 在P2WPKH和P2WSH中只接受压缩公钥.[BIP143](https://link.jianshu.com/?t=https://github.com/bitcoin/bips/blob/master/bip-0143.mediawiki%23Restrictions_on_public_key_type" \t "https://www.jianshu.com/p/_blank)
2. 在P2WSH中OP\_IF/NOTIF的参数必须是最小的.
3. 如果OP\_CHECKSIG或OP\_CHECKMULTISIG失败，签名必须是空的字节向量.(当前的segregated witness脚本和P2WSH，请看[BIP146](https://link.jianshu.com/?t=https://github.com/bitcoin/bips/blob/master/bip-0146.mediawiki" \t "https://www.jianshu.com/p/_blank))

## **示例**

### **P2WPKH**

接下来的例子是0号版本的P2WPKH

witness: <signature> <pubkey>

scriptSig: (empty)

scriptPubKey: 0 <20-byte-key-hash>

(0x0014{20-byte-key-hash})

'0'在锁定脚本中标识接下来的push数据是一个版本位0的witness program。witness program的长度标识脚本是一个P2WPKH类型。witness必须恰好包含2项，在witness中公钥的hash160必须匹配witness program。

签名被如此验证：<signature> <pubkey> CHECKSIG

与传统的P2PKH输出相比，P2WPKH在scriptPubKey中占据少3个字节，并且从签名脚本中转移签名和公钥至witness字段。

### **P2WPKH 嵌套在 BIP16 P2SH中**

witness: <signature> <pubkey>

scriptSig: <0 <20-byte-key-hash>>

(0x160014{20-byte-key-hash})

scriptPubKey: HASH160 <20-byte-script-hash> EQUAL

(0xA914{20-byte-script-hash}87)

在签名脚本中仅包含Hash160，与锁定脚本中的20字节的脚本哈希进行比较，并被解释为：0 <20-byte-key-hash>.

与以前的例子比较，scriptPubKey 大1字节，并且签名脚本大23字节。尽管一个嵌套的witness program是不太高效，但它的支付地址是完全透明的，并且向后兼容比特币0.6.0以后的所有版本。

### **P2WSH**

接下来的示例是1-of-2的多重签名，版本号0

witness: 0 <signature1> <1 <pubkey1> <pubkey2> 2 CHECKMULTISIG>

scriptSig: (empty)

scriptPubKey: 0 <32-byte-hash>

(0x0020{32-byte-hash})

在scriptPubKey 中的'0'标识接下来的push操作是一个版本号为0的witness program。witness program的长度标识是P2WSH类型。在witness中最后一项被弹出，并被SHA256后，与在scriptPubKey中的32字节进行比较，和反序列化。1 <pubkey1> <pubkey2> 2 CHECKMULTISIG

脚本使用witness中其余的数据进行执行。  
0 <signature1> 1 <pubkey1> <pubkey2> 2 CHECKMULTISIG

P2WSH 允许最大字节脚本字节为10,000，520字节的限制被绕过。

scriptPubKey 占据34字节，而不是P2SH中的23字节。增加字节提升了可能的碰撞攻击的安全性，因为2\*\*80是不再可行。可花费的脚本与BIP16中的交易输出相同，但被已到了witness字段中。

### **P2WSH nested in BIP16 P2SH**

接下来的示例是1-of-2的多重签名，版本号0，但被嵌套在BIP16的P2SH输出中。

witness: 0 <signature1> <1 <pubkey1> <pubkey2> 2 CHECKMULTISIG>

scriptSig: <0 <32-byte-hash>>

(0x220020{32-byte-hash})

scriptPubKey: HASH160 <20-byte-hash> EQUAL

(0xA914{20-byte-hash}87)

对签名脚本中的唯一项进行HASH160,然后与锁定脚本中的20字节进行比较，被解释为 0 <32-byte-hash>.

P2WSH的witnessScript接下来如上述示例继续执行。

与以前的示例比较，锁定脚本是小11个字节(降低了安全)；但是，它在签名脚本中要求了35字节的数据。

### **可拓展的承诺结构(Extensible commitment structure)**

在coinbase交易中新的承诺结构是哈希witness root hash和witness reserved value。witness reserved value当前没有共识含义，但是在未来的软分叉中允许成为新的共识结构。例如：如果在未来要求一个新的严格共识结构，在coinbase中的承诺结构将变为：Double-SHA256(Witness root hash|Hash(new commitment|witness reserved value))

为了向后兼容，Hash(new commitment|witness reserved value)将移动到coinbase的 witness字段，并且witness reserved value将通过软分叉被记录在另一个地方。任何数量的新的共识结构都可以通过这种方式添加。

任何对于比特币不严格共识的承诺结构，如合并挖矿，一定不能使用witness reserved value字段来保留升级比特币共识协议的能力。

### **无信任的未确认交易依赖链**

Segregated witness 从根本上修正了交易延展性问题，以一种无信任的方式构建未确认的交易依赖链。

两个成员，Alice和Bob，可能同意发送一定数量的比特币到2-of-2的多重签名输出(资金交易)。不需要签名这个资金交易，他们就可以创建另一个交易，时间锁定在未来，花费2-of-2的多重签名输出至第三方(花费交易)。ALice和Bob将签署花费交易，并交换签名。验证通过签名后，他们将签署并提交现金交易至块链。不需要进一步的操作，花费交易将在时间戳到达之后被确认，并根据原始合同释放资金。它也保留了在时间戳之间撤销原始合同的灵活性，通过使用一个更短的时间戳的花费交易，但是这种情况只能在双方达成共识的情况下发生。

BIP62 的设置不可能修复交易延展性，因为花费交易不可能在双方没有签署资金交易的情况下创建。如果Alice比Bob早一步展示资金交易签名，Bob可以无限期的锁定资金，而无需签署任何花费交易。

未确认的交易依赖链是更复杂的支付网络基本组成部分，例如：全双工微型支付通道和闪电网络，可能隐式的极大提升比特币系统的可拓展性和效率。

## **未来的拓展**

### **用于SPV节点的紧凑型欺诈证明**

比特币当前仅含有两种安全模式。在系统中，用户要么运行全节点使用所有的规则验证每个区块，或一个SPV客户端仅验证头部作为一些交易的发布证明。比特币白皮书建议SPV客户端可以接收来自全节点的警告，即当全节点检测到一个无效的区块，提示SPV节点去下载这个有问题的区块和交易去验证。然而，这种方法可能会成为DOS攻击的载体，因为可以几乎没有成本的产生警告。警告必须具有紧凑但确定性的欺诈证明。

在当前的比特币协议中，对于几乎所有的规则，都能产生紧凑型欺诈证明，除了特别少的一些例外：

1. 当没有展示整个区块和它的所有交易输入，则不可能证明矿工在coinbase交易输出中引入了太多的比特币。
2. 当没有展示整个区块(和所有的交易输入操作码)，则不可能证明违反了块的指定限制(例如：字节和操作码数量限制)。
3. 没有展示块链中的所有的交易ID显示在创建的区块中，则不能证明花费了不存在交易输入。

可以提交额外的witness数据，以允许SPV节点可以快速验证无效的区块的简短证明。

1. 交易费的总和可以用来建立简短的证明，证明矿工没有在coinbase交易中添加额外的交易费。类似于区块的大小和操作码计数限制。
2. 可以提供交易输入所花费的输出的反向链接。反向链接由块的哈希和一个偏移量组成，请客户端可以非常容易的查询和检查验证交易输出的存在。  
   这些承诺结构可以通过软分叉被包含在可拓展的承诺结构中，并且对于不理解新规则的节点将是透明的。

### **新的脚本系统**

因为一个版本字节在witness program之前被push, 并且使用未知版本号的program被认为是anyone-can-spend脚本，所有可以通过软分叉引入任何新的脚本系统。witness结构不会被任何已存在的脚本语义和限制所限制，特别是520字节的push限制，因此允许任意大的脚本和签名。

例如：包含Schnorr签名的新脚本系统将显著减少多签名交易的字节；包含Schnorr签名的脚本是具有抗量子计算的，以及Merklized抽象语义树允许在非常复杂的条件脚本上含有非常紧凑的witness数据。

### **每个输入的锁定时间和相对锁定时间**

在交易中仅含有一个nLockTime字段，所有的交易输入飞享相同的值。[BIP68](https://link.jianshu.com/?t=https://github.com/bitcoin/bips/blob/master/bip-0068.mediawiki" \t "https://www.jianshu.com/p/_blank)使用nSequence字段标识相对锁定时间，然而，锁定时间和和决议有限制。

使用软分叉，可以引入一个分离的witness结构，允许每个交易输入有自己的锁定时间和相当锁定时间，并且新的脚本系统可以签名和操作新的数据(类似于[BIP65](https://link.jianshu.com/?t=https://github.com/bitcoin/bips/blob/master/bip-0065.mediawiki" \t "https://www.jianshu.com/p/_blank)和[BIP112](https://link.jianshu.com/?t=https://github.com/bitcoin/bips/blob/master/bip-0112.mediawiki" \t "https://www.jianshu.com/p/_blank))。

## **向后兼容**

作为一个软分叉，旧版本的软件将没有任何变化的继续执行。未升级的节点，将不能看到并验证witness数据，并且认为所有的witness program都是anyone-can-spend脚本(除了一些边缘的案例，witness program等于0，该脚本执行肯定失败)。钱包应当一直警惕anyone-can-spend脚本，并对他们持怀疑态度。所有未升级的节点强烈建议去升级，以便可以采用新功能。

未升级的钱包可以做什么：

1. 从已升级和未升级的钱包接收比特币
2. 发送比特币给未升级的钱包，并且升级传统P2PKH地址的钱包(没有隔离见证交易的好处)
3. 使用P2SH地址发送比特币给升级的钱包
4. 通过[BIP70](https://link.jianshu.com/?t=https://github.com/bitcoin/bips/blob/master/bip-0070.mediawiki" \t "https://www.jianshu.com/p/_blank) 支付协议，使用本地隔离程序发送比特币给升级的钱包

未升级的钱包不可以做什么：

1. 验证隔离见证交易。它假设这样的一个交易是一直有效的。

## **部署**

该BIP通过BIP9的版本字节和SegWit名称，使用bit 1的位置进行部署。  
主网络的部署时间(都为BIP9的中位数时间)：开始：2016.11.15; 结束：2017.11.15  
测试网路的部署时间(都为BIP9的中位数时间): 开始：2016.05.01； 结束：2017.05.01

## **脚注：**

案例: 携带OP\_0以及40个非0字节PUSH数据的锁定脚本，由于不正确的字节长度，将导致验证失败。然而，携带OP\_0以及41个非0字节的PUSH数据将通过，因为它不被认为是一个witness程序。

为了向后兼容，对于从0到16的任何版本字节，如果这个witness程序的CastToBool的值为0，这个脚本必须失败。然而，这样的哈希是一次成功的对哈希函数的原项攻击，并且风险可以忽略。

合理使用单一复合约束，替代两个分开的限制：1M的基本数据和3M witness数据；使用两个单独的限制将使挖矿和交易费预估几乎不可能。矿工将要解决一个复杂的非线性优化问题:以找到一组符合两者的最大交易费，且钱包无法知道需要支付多少交易费，因为此时的交易费依赖于当矿工使用交易产生区块时，这两个条件哪个条件受到最高约束。这种方法的另一个问题是freeloading。一旦一组交易的基本数据达到1M限制，仅通过最小化增加费用就可以给witness字段添加额外高达3M的数据。在这种情况下，额外witness空间的边际成本实际变为0.

## **引用实现**

[https://github.com/bitcoin/bitcoin/pull/8149](https://link.jianshu.com/?t=https://github.com/bitcoin/bitcoin/pull/8149" \t "https://www.jianshu.com/p/_blank)