```
BIP: 178
Layer: Applications
Title: Version Extended WIF
Author: Karl-Johan Alm <karljohan-alm@garage.co.jp>
Comments-Summary: Discouraged for implementation (one person)
Comments-URI: https://github.com/bitcoin/bips/wiki/Comments:BIP-0178
Status: Draft
Type: Standards Track
Created: 2018-04-04
License: CC0-1.0
```

Abstract

An extension to the Wallet Import Format (WIF) to specify what kind of bitcoin address the private key corresponds to.

Motivation

There are several types of bitcoin addresses which can all be associated with a given private key: P2PKH (legacy `1...` format), P2SH-P2WPKH (SegWit public key inside P2SH), P2WPKH (bech32), etc.

While private keys have a 1-byte suffix indicating whether the corresponding public key is compressed (`0x01`) or not (through suffix absence), there is no way of knowing what kind of bitcoin address were associated with the private key. As a result, when importing a private key, the wallet has to assume all kinds, and keep track of each possible alternative.

By extending the suffix, we can specify what kind of bitcoin address was associated with a given private key.

Specification

Currently, private keys are stored as a uint256 (private key data) followed by an optional uint8 (compressed flag). The latter is extended to specify the address types:

Value	Type	Compr	Clarification
No suffix	P2PKH_UNCOMPRESSED	No	Uncompressed legacy public key. Unknown public key fo
`0x01`	P2PKH_COMPRESSED	Yes	Compressed legacy public key. Unknown public key form
`0x10`	P2PKH	Yes	Compressed legacy public key. Legacy public key format
`0x11`	P2WPKH	Yes	Bech32 format (native Segwit)
`0x12`	P2WPKH_P2SH	Yes	Segwit nested in BIP16 P2SH (`3...`)

When a wallet imports a private key, it will have two outcomes:

- the key is using one of the legacy types, in which case all types must be accounted for
- the key is using one of the extended types, in which case the wallet need only track the specific corresponding address

Note: the difference between ‘0x01‘ and ‘0x10‘ is that the former can correspond to any of the types above, whereas the latter *only* corresponds to a P2PKH (legacy non-segwit).

Compatibility

This proposal is not backwards compatible, in that software that does not recognize the new types will not understand the compressed flag. It would be trivial to change this, by keeping the 'uncompressed' state as it is (no suffix) and changing 'compressed' to be 'anything not 0', as opposed to 'the value 1'.

The proposal *is* backwards compatible in that new wallet software will always understand the old WIF format, however. It will, as it does today, assume that any kind of bitcoin address is possible, and will have to track all of them, as it has to today.

Acknowledgements

This BIP is based on the initial proposal by Thomas Voegtlin (thomasv at electrum dot org) on the Bitcoin Dev mailing list[1] and the Electrum 3.0 implementation[2]

Reference implementation

There is a partial implementation which adds, but does not use, the types described in this BIP here: https://github.com/bitcoin/bitcoin/pull/12869

References

Copyright

[1] https://lists.linuxfoundation.org/pipermail/bitcoin-dev/2017-September/015007.html

[2] https://github.com/spesmilo/electrum/blob/82e88cb89df35288b80dfdbe071da74247351251/RELEASE-NOTES#L95-L108