```
BIP: 382
Layer: Applications
Title: Segwit Output Script Descriptors
Author: Pieter Wuille <pieter@wuille.net>
        Andrew Chow <andrew@achow101.com>
Comments-Summary: No comments yet.
Comments-URI: https://github.com/bitcoin/bips/wiki/Comments:BIP-0382
Status: Draft
Type: Informational
Created: 2021-06-27
License: BSD-2-Clause
```

Abstract

This document specifies `wpkh()`, and `wsh()` output script descriptors. `wpkh()` descriptors take a key and produces a P2WPKH output script. `wsh()` descriptors take a script and produces a P2WSH output script.

Copyright

Motivation

Segregated Witness added 2 additional standard output script formats: P2WPKH and P2WSH. These expressions allow specifying those formats as a descriptor.

Specification

Two new script expressions are defined: `wpkh()`, and `wsh()`.

`wpkh()`

The `wpkh(KEY)` expression can be used as a top level expression, or inside of a `sh()` descriptor. It takes a single key expression as an argument and produces a P2WPKH output script. Only keys which are/has compressed public keys can be contained in a `wpkh()` expression.

The output script produced is:

```
OP_0 <KEY_hash160>
```

`wsh()`

The `wsh(SCRIPT)` expression can be used as a top level expression, or inside of a `sh()` descriptor. It takes a single script expression as an argument and produces a P2WSH output script. `wsh()` expressions also create a witnessScript which is required in order to spend outputs which use its output script. This

redeemScript is the output script produced by the SCRIPT argument to wsh(). Any key expression found in any script expression contained by a wsh() expression must only produce compressed public keys.

The output script produced is:

OP_0 <SCRIPT_sha256>

Test Vectors

Valid descriptors followed by the scripts they produce. Descriptors involving derived child keys will have the 0th, 1st, and 2nd scripts listed.

- wpkh(L4rK1yDtCWekvXuE6oXD9jCYfFNV2cWRpVuPLBcCU2z8TrisoyY1)
  - 00149a1c78a507689f6f54b847ad1cef1e614ee23f1e
- wpkh(03a34b99f22c790c4e36b2b3c2c35a36db06226e41c692fc82b8b56ac1c540c5bd)
  - 00149a1c78a507689f6f54b847ad1cef1e614ee23f1e
- wpkh([ffffffff/13']xprv9vHkqa6EV4sPZHYqZznhT2NPtPCjKuDKGY38FBWLvgaDx45zo9WQRUT3dKYnjwih
  - 0014326b2249e3a25d5dc60935f044ee835d090ba859
- wpkh([ffffffff/13']xpub69H7F5d8KSRgmmdJg2KhpAK8SR3DjMwAdkxj3ZuxV27CprR9LgpeyGmXUbC6wb7E
  - 0014326b2249e3a25d5dc60935f044ee835d090ba859
  - 0014af0bd98abc2f2cae66e36896a39ffe2d32984fb7
  - 00141fa798efd1cbf95cebf912c031b8a4a6e9fb9f27
- sh(wpkh(xprv9s21ZrQH143K3QTDL4LXw2F7HEK3wJUD2nW2nRk4stbPy6cq3jPPqjiChkVvvNKmPGJxWUtg6Ln
  - a9149a4d9901d6af519b2a23d4a2f51650fcba87ce7b87
  - a914bed59fc0024fae941d6e20a3b44a109ae740129287
  - a9148483aa1116eb9c05c482a72bada4b1db24af654387
- sh(wpkh(xpub661MyMwAqRbcFtXgS5sYJABqqG9YLmC4Q1Rdap9gSE8NqtwybGhePY2gZ29ESFjqJoCu1Rupje8
  - a9149a4d9901d6af519b2a23d4a2f51650fcba87ce7b87
  - a914bed59fc0024fae941d6e20a3b44a109ae740129287
  - a9148483aa1116eb9c05c482a72bada4b1db24af654387
- wsh(pkh(L4rK1yDtCWekvXuE6oXD9jCYfFNV2cWRpVuPLBcCU2z8TrisoyY1))
  - 0020338e023079b91c58571b20e602d7805fb808c22473cbc391a41b1bd3a192e75b
- wsh(pkh(03a34b99f22c790c4e36b2b3c2c35a36db06226e41c692fc82b8b56ac1c540c5bd))
  - 0020338e023079b91c58571b20e602d7805fb808c22473cbc391a41b1bd3a192e75b
- wsh(pk(L4rK1yDtCWekvXuE6oXD9jCYfFNV2cWRpVuPLBcCU2z8TrisoyY1))
  - 00202e271faa2325c199d25d22e1ead982e45b64eeb4f31e73dbdf41bd4b5fec23fa
- wsh(pk(03a34b99f22c790c4e36b2b3c2c35a36db06226e41c692fc82b8b56ac1c540c5bd))
  - 00202e271faa2325c199d25d22e1ead982e45b64eeb4f31e73dbdf41bd4b5fec23fa
- sh(wsh(pkh(L4rK1yDtCWekvXuE6oXD9jCYfFNV2cWRpVuPLBcCU2z8TrisoyY1)))
  - a914b61b92e2ca21bac1e72a3ab859a742982bea960a87
- sh(wsh(pkh(03a34b99f22c790c4e36b2b3c2c35a36db06226e41c692fc82b8b56ac1c540c5bd)))
  - a914b61b92e2ca21bac1e72a3ab859a742982bea960a87

Invalid descriptors with descriptions

- Uncompressed public key in wpkh(): wpkh(5KYZdUEo39z3FPrtuX2QbbwGnNP5zTd7yyr2SC1j299sBCnWjss)
- Uncompressed public key in wpkh(): sh(wpkh(5KYZdUEo39z3FPrtuX2QbbwGnNP5zTd7yyr2SC1j299sBCnWj

- Uncompressed public key in `wpkh()`: `wpkh(04a34b99f22c790c4e36b2b3c2c35a36db06226e41c692fc82b8`
- Uncompressed public key in `wpkh()`: `sh(wpkh(04a34b99f22c790c4e36b2b3c2c35a36db06226e41c692fc8`
- Uncompressed public keys under `wsh()`: `wsh(pk(5KYZdUEo39z3FPrtuX2QbbwGnNP5zTd7yyr2SC1j299sBCn`
- Uncompressed public keys under `wsh()`: `wsh(pk(04a34b99f22c790c4e36b2b3c2c35a36db06226e41c692f`
- `wpkh()` nested in `wsh()`: `wsh(wpkh(03a34b99f22c790c4e36b2b3c2c35a36db06226e41c692fc82b8b56ac`
- `wsh()` nested in `wsh()`: `wsh(wsh(pkh(03a34b99f22c790c4e36b2b3c2c35a36db06226e41c692fc82b8b56a`
- `wsh()` nested in `wsh()`: `sh(wsh(wsh(pkh(03a34b99f22c790c4e36b2b3c2c35a36db06226e41c692fc82b8b`
- Script in `wpkh()`: `wpkh(wsh(pkh(03a34b99f22c790c4e36b2b3c2c35a36db06226e41c692fc82b8b56ac1c`
- Key in `wsh()`: `wsh(03a34b99f22c790c4e36b2b3c2c35a36db06226e41c692fc82b8b56ac1c540c5bd)`

Backwards Compatibility

`wpkh()`, and `wsh()` descriptors use the format and general operation specified in 380. As these are a wholly new descriptors, they are not compatible with any implementation. However the scripts produced are standard scripts so existing software are likely to be familiar with them.

Reference Implementation

`wpkh()`, and `wsh()` descriptors have been implemented in Bitcoin Core since version 0.17.