```
BIP: 386
Layer: Applications
Title: tr() 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-0386
Status: Draft
Type: Informational
Created: 2021-06-27
License: BSD-2-Clause
```

Abstract

This document specifies `tr()` output script descriptors. `tr()` descriptors take a key and optionally a tree of scripts and produces a P2TR output script.

Copyright

Motivation

Taproot added one additional standard output script format: P2TR. These expressions allow specifying those formats as a descriptor.

Specification

A new script expression is defined: `tr()`. A new expression is defined: Tree Expressions

Tree Expression

A Tree Expression (denoted `TREE`) is an expression which represents a tree of scripts. The way the tree is represented in an output script is dependent on the higher level expressions.

A Tree Expression is:

- Any Script Expression that is allowed at the level this Tree Expression is in.
- A pair of Tree Expressions consisting of:
  - An open brace `{`
  - A Tree Expression
  - A comma `,`
  - A Tree Expression
  - A closing brace `}`

`tr()`

The `tr(KEY)` or `tr(KEY, TREE)` expression can only be used as a top level expression. All key expressions under any `tr()` expression must create x-only public keys.

`tr(KEY)` takes a single key expression as an argument and produces a P2TR output script which does not have a script path. Each key produced by the key expression is used as the internal key of a P2TR output as specified by BIP 341. Specifically, "If the spending conditions do not require a script path, the output key should commit to an unspendable script path instead of having no script path. This can be achieved by computing the output key point as $Q = P + int(hash_{TapTweak}(bytes(P)))G$."

```
internal_key:       lift_x(KEY)
32_byte_output_key: internal_key + int(HashTapTweak(bytes(internal_key)))G
scriptPubKey:       OP_1 <32_byte_output_key>
```

`tr(KEY, TREE)` takes a key expression as the first argument, and a tree expression as the second argument and produces a P2TR output script which has a script path. The keys produced by the first key expression are used as the internal key as specified by BIP 341. The Tree expression becomes the Taproot script tree as described in BIP 341. A merkle root is computed from this tree and combined with the internal key to create the Taproot output key.

```
internal_key:       lift_x(KEY)
merkle_root:        HashTapBranch(TREE)
32_byte_output_key: internal_key + int(HashTapTweak(bytes(internal_key) || merkle_root))G
scriptPubKey:       OP_1 <32_byte_output_key>
```

Modified Key Expression

Key Expressions within a `tr()` expression must only create x-only public keys. Uncompressed public keys are not allowed, but compressed public keys would be implicitly converted to x-only public keys. The keys derived from extended keys must be serialized as x-only public keys. An additional key expression is defined only for use within a `tr()` descriptor:

- A 64 hex character string representing an x-only public key

Test Vectors

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

- `tr(a34b99f22c790c4e36b2b3c2c35a36db06226e41c692fc82b8b56ac1c540c5bd)`
  - `512077aab6e066f8a7419c5ab714c12c67d25007ed55a43cadcacb4d7a970a093f11`
- `tr(L4rK1yDtCWekvXuE6oXD9jCYfFNV2cWRpVuPLBcCU2z8TrisoyY1)`
  - `512077aab6e066f8a7419c5ab714c12c67d25007ed55a43cadcacb4d7a970a093f11`

- `tr(xprvA1RpRA33e1JQ7ifknakTFpgNXPmW2YvmhqLQYMmrj4xJXXWYpDPS3xz7iAxn8L39njGVyuoseXzU6rcx`
  - `512078bc707124daa551b65af74de2ec128b7525e10f374dc67b64e00ce0ab8b3e12`
  - `512001f0a02a17808c20134b78faab80ef93ffba82261ccef0a2314f5d62b6438f11`
  - `512021024954fcec88237a9386fce80ef2ced5f1e91b422b26c59ccfc174c8d1ad25`
- `tr(a34b99f22c790c4e36b2b3c2c35a36db06226e41c692fc82b8b56ac1c540c5bd,pk(669b8afcec803a0d`
  - `512017cf18db381d836d8923b1bdb246cfcd818da1a9f0e6e7907f187f0b2f937754`
- `tr(a34b99f22c790c4e36b2b3c2c35a36db06226e41c692fc82b8b56ac1c540c5bd,{pk(xprvA2JDeKCSNNZ`
  - `512071fff39599a7b78bc02623cbe814efebf1a404f5d8ad34ea80f213bd8943f574`

Invalid Descriptors

- Uncompressed private key: `tr(5kyzdueo39z3fprtux2qbbwgnnp5ztd7yyr2sc1j299sbcnwjss)`
- Uncompressed public key: `tr(04a34b99f22c790c4e36b2b3c2c35a36db06226e41c692fc82b8b56ac1c540`
- `tr()` nested in `wsh`: `wsh(tr(a34b99f22c790c4e36b2b3c2c35a36db06226e41c692fc82b8b56ac1c540c5bd`
- `tr()` nested in `sh`: `sh(tr(a34b99f22c790c4e36b2b3c2c35a36db06226e41c692fc82b8b56ac1c540c5bd))`
- `pkh()` nested in `tr`: `tr(a34b99f22c790c4e36b2b3c2c35a36db06226e41c692fc82b8b56ac1c540c5bd, pkh(L4rK1yDtCWekvXuE6oXD9jCYfFNV2cWRpVuPLBcCU2z8TrisoyY1))`

Backwards Compatibility

`tr()` descriptors use the format and general operation specified in 380. As these are a set of 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.

Tree Expressions are largely incompatible with existing script expressions due to the restrictions in those expressions. As of 2021-06-27, the only allowed script expression that can be used in a tree expression is `pk()`. However there will be future BIPs that specify script expressions that can be used in tree expressions.

Reference Implementation

`tr()` descriptors have been implemented in Bitcoin Core since version 22.0.