```
BIP: 371
Layer: Applications
Title: Taproot Fields for PSBT
Author: Andrew Chow <andrew@achow101.com>
Comments-Summary: No comments yet.
Comments-URI: https://github.com/bitcoin/bips/wiki/Comments:BIP-0371
Status: Draft
Type: Standards Track
Created: 2021-06-21
License: BSD-2-Clause
```

Introduction

Abstract

This document proposes additional fields for BIP 174 PSBTv0 and BIP 370 PSBTv2 that allow for BIP 340/341/342 Taproot data to be included in a PSBT of any version. These will be fields for signatures and scripts that are relevant to the creation of Taproot inputs.

Copyright

This BIP is licensed under the 2-clause BSD license.

Motivation

BIPs 340, 341, and 342 specify Taproot which provides a wholly new way to create and spend Bitcoin outputs. The existing PSBT fields are unable to support Taproot due to the new signature algorithm and the method by which scripts are embedded inside of a Taproot output. Therefore new fields must be defined to allow PSBTs to carry the information necessary for signing Taproot inputs.

Specification

The new per-input types are defined as follows:

Name		
Taproot Key Spend Signature	`PSBT_IN_TAP_KEY_SIG = 0x13`	None
Taproot Script Spend Signature	`PSBT_IN_TAP_SCRIPT_SIG = 0x14`	
Taproot Leaf Script	`PSBT_IN_TAP_LEAF_SCRIPT = 0x15`	
Taproot Key BIP 32 Derivation Path	`PSBT_IN_TAP_BIP32_DERIVATION = 0x16`	`<32 byte xonlypubkey>`
Taproot Internal Key	`PSBT_IN_TAP_INTERNAL_KEY = 0x17`	None
Taproot Merkle Root	`PSBT_IN_TAP_MERKLE_ROOT = 0x18`	None

The new per-output types are defined as follows:

Name		
Taproot Internal Key	`PSBT_OUT_TAP_INTERNAL_KEY = 0x05`	None
Taproot Tree	`PSBT_OUT_TAP_TREE = 0x06`	None
Taproot Key BIP 32 Derivation Path	`PSBT_OUT_TAP_BIP32_DERIVATION = 0x07`	`<32 byte xonlypubkey`

UTXO Types

BIP 174 recommends using `PSBT_IN_NON_WITNESS_UTXO` for all inputs because of potential attacks involving an updater lying about the amounts in an output. Because a Taproot signature will commit to all of the amounts and output scripts spent by the inputs of the transaction, such attacks are prevented as any such lying would result in an invalid signature. Thus Taproot inputs can use just `PSBT_IN_WITNESS_UTXO`.

Compatibility

These are simply new fields added to the existing PSBT format. Because PSBT is designed to be extensible, old software will ignore the new fields.

Test Vectors

The following are invalid PSBTs:

- Case: PSBT With `PSBT_IN_TAP_INTERNAL_KEY` key that is too long (incorrectly serialized as compressed DER)
  - Bytes in Hex:
-
  - Base64 String:
    `cHNidP8BAHECAAAAASdOSrq/MCf+DWzyOpbu4u+xiO9SMBlUWFiD5ptmJLJCAAAAAAD/////Anh8AQAAAA`
- Case: PSBT With `PSBT_KEY_PATH_SIG` signature that is too short
  - Bytes in Hex:
    `<70736274ff0100710200000001277744ababf3027fe0d6cf23a96eee2efb188ef52301954585883e69b`
  - Base64 String:
    `cHNidP8BAHECAAAAASdOSrq/MCf+DWzyOpbu4u+xiO9SMBlUWFiD5ptmJLJCAAAAAAD/////Anh8AQAAAA`
- Case: PSBT With `PSBT_KEY_PATH_SIG` signature that is too long
  - Bytes in Hex:

[1] **Why is there no key data for `PSBT_IN_TAP_KEY_SIG`**The signature in a key path spend corresponds directly with the pubkey provided in the output script. Thus it is not necessary to provide any metadata that attaches the key path spend signature to a particular pubkey.

[2] **Why is the internal key provided?**The internal key is not necessarily the same key as in the Taproot output script. BIP 341 recommends tweaking the key with the hash of itself. It may be necessary for signers to know what the internal key actually is so that they are able to determine whether an input can be signed by it.

<70736274ff0100710200000001277744ababf3027fe0d6cf23a96eee2efb188ef52301954585883e69b
  - Base64 String:
cHNidP8BAHECAAAAASd0Srq/MCf+DWzyOpbu4u+xiO9SMBlUWFiD5ptmJLJCAAAAAAD/////Anh8AQAAAA

- Case: PSBT With PSBT_IN_TAP_BIP32_DERIVATION key that is too long (incorrectly serialized as compressed DER)
  - Bytes in Hex:
<70736274ff0100710200000001277744ababf3027fe0d6cf23a96eee2efb188ef52301954585883e69b
  - Base64 String:
cHNidP8BAHECAAAAASd0Srq/MCf+DWzyOpbu4u+xiO9SMBlUWFiD5ptmJLJCAAAAAAD/////Anh8AQAAAA

- Case: PSBT With PSBT_OUT_TAP_INTERNAL_KEY key that is too long (incorrectly serialized as compressed DER)
  - Bytes in Hex:
70736274ff01007d020000000127744ababf3027fe0d6cf23a96eee2efb188ef52301954585883e69b6
  - Base64 String:
cHNidP8BAH0CAAAAASd0Srq/MCf+DWzyOpbu4u+xiO9SMBlUWFiD5ptmJLJCAAAAAAD/////Aoh7AQAAAA

- Case: PSBT With PSBT_OUT_TAP_BIP32_DERIVATION key that is too long (incorrectly serialized as compressed DER)
  - Bytes in Hex:
70736274ff01007d020000000127744ababf3027fe0d6cf23a96eee2efb188ef52301954585883e69b6
  - Base64 String:
cHNidP8BAH0CAAAAASd0Srq/MCf+DWzyOpbu4u+xiO9SMBlUWFiD5ptmJLJCAAAAAAD/////Aoh7AQAAAA

- Case: PSBT With PSBT_IN_TAP_SCRIPT_SIG key that is too long (incorrectly serialized as compressed DER)
  - Bytes in Hex:
70736274ff01005e02000000019bd48765230bf9a72e662001f972556e54f0c6f97feb56bcb5600d817
  - Base64 String:
cHNidP8BAF4CAAAAAZvUh2UjC/mnLmYgAflyVW5U8Mb5f+tWvLVgDYF/aZUmAQAAAAD/////AUjmBSoBAAA

- Case: PSBT With PSBT_IN_TAP_SCRIPT_SIG signature that is too long
  - Bytes in Hex:
70736274ff01005e02000000019bd48765230bf9a72e662001f972556e54f0c6f97feb56bcb5600d817
  - Base64 String:
cHNidP8BAF4CAAAAAZvUh2UjC/mnLmYgAflyVW5U8Mb5f+tWvLVgDYF/aZUmAQAAAAD/////AUjmBSoBAAA

- Case: PSBT With PSBT_IN_TAP_SCRIPT_SIG signature that is too short
  - Bytes in Hex:
70736274ff01005e02000000019bd48765230bf9a72e662001f972556e54f0c6f97feb56bcb5600d817
  - Base64 String:
cHNidP8BAF4CAAAAAZvUh2UjC/mnLmYgAflyVW5U8Mb5f+tWvLVgDYF/aZUmAQAAAAD/////AUjmBSoBAAA

- Case: PSBT With PSBT_IN_TAP_LEAF_SCRIPT Control block that is too long
  - Bytes in Hex:
70736274ff01005e02000000019bd48765230bf9a72e662001f972556e54f0c6f97feb56bcb5600d817
  - Base64 String:

cHNidP8BAF4CAAAAAZvUh2UjC/mnLmYgAflyVW5U8Mb5f+tWvLVgDYF/aZUmAQAAAAD/////AUjmBSoBAAA

- Case: PSBT With PSBT_IN_TAP_LEAF_SCRIPT Control block that is too short
  - Bytes in Hex:
    70736274ff01005e02000000019bd48765230bf9a72e662001f972556e54f0c6f97feb56bcb5600d817
  - Base64 String:
    cHNidP8BAF4CAAAAAZvUh2UjC/mnLmYgAflyVW5U8Mb5f+tWvLVgDYF/aZUmAQAAAAD/////AUjmBSoBAAA

The following are valid PSBTs:

- Case: PSBT with one P2TR key only input with internal key and its derivation path
  - Bytes in Hex:
    70736274ff0100520200000000127744ababf3027fe0d6cf23a96eee2efb188ef52301954585883e69b6
  - Base64 String:
    cHNidP8BAFICAAAAASd0Srq/MCf+DWzyOpbu4u+xiO9SMBlUWFiD5ptmJLJCAAAAAAD/////AUjmBSoBAAA
- Case: PSBT with one P2TR key only input with internal key, its derivation path, and signature
  - Bytes in Hex:
    70736274ff0100520200000000127744ababf3027fe0d6cf23a96eee2efb188ef52301954585883e69b6
  - Base64 String:
    cHNidP8BAFICAAAAASd0Srq/MCf+DWzyOpbu4u+xiO9SMBlUWFiD5ptmJLJCAAAAAAD/////AUjmBSoBAAA
- Case: PSBT with one P2TR key only output with internal key and its derivation path
  - Bytes in Hex:
    70736274ff01005e020000000127744ababf3027fe0d6cf23a96eee2efb188ef52301954585883e69b6
  - Base64 String:
    cHNidP8BAF4CAAAAASd0Srq/MCf+DWzyOpbu4u+xiO9SMBlUWFiD5ptmJLJCAAAAAAD/////AUjmBSoBAAA
- Case: PSBT with one P2TR script path only input with dummy internal key, scripts, derivation paths for keys in the scripts, and merkle root
  - Bytes in Hex:
    70736274ff01005e02000000019bd48765230bf9a72e662001f972556e54f0c6f97feb56bcb5600d817
  - Base64 String:
    cHNidP8BAF4CAAAAAZvUh2UjC/mnLmYgAflyVW5U8Mb5f+tWvLVgDYF/aZUmAQAAAAD/////AUjmBSoBAAA
- Case: PSBT with one P2TR script path only output with dummy internal key, taproot tree, and script key derivation paths
  - Bytes in Hex:
    70736274ff01005e020000000127744ababf3027fe0d6cf23a96eee2efb188ef52301954585883e69b6
  - Base64 String:
    cHNidP8BAF4CAAAAASd0Srq/MCf+DWzyOpbu4u+xiO9SMBlUWFiD5ptmJLJCAAAAAAD/////AUjmBSoBAAA
- Case: PSBT with one P2TR script path only input with dummy internal key, scripts, script key derivation paths, merkle root, and script path signatures
  - Bytes in Hex:

70736274ff01005e02000000019bd48765230bf9a72e662001f972556e54f0c6f97feb56bcb5600d817

– Base64 String:
cHNidP8BAF4CAAAAAZvUh2UjC/mnLmYgAflyVW5U8Mb5f+tWvLVgDYF/aZUmAQAAAAD/////AUjmBSoBAAA

Rationale

Reference implementation

The reference implementation of the PSBT format is available at https://github.com/achow101/bitcoin/tree/taproot-psbt.

Acknowledgements

TBD