

BIP380 Descriptor Key Expressions: Hex, WIF, and BIP32

October 24, 2026 · Document #407 · bip39-phrase.com

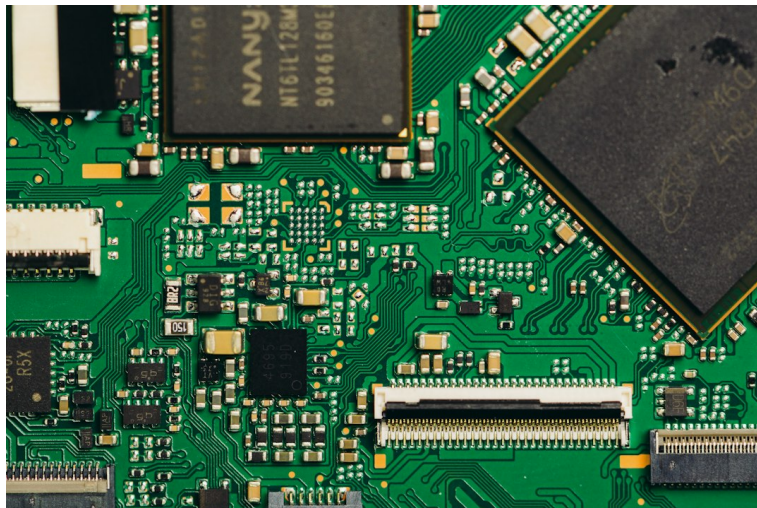


Photo by [Albert Stoynov](#) on [Unsplash](#)

Learn how BIP380 defines key expressions in Bitcoin output descriptors, covering hex keys, WIF, xpubs, derivation paths, and script context rules.

Contents

Structure of a BIP380 Key Expression	2
Supported Key Formats	2
Script Contexts and Compatibility Constraints	2
Private Keys in Descriptors and Security	3
FAQ	4
Sources	4
BIP39 format examples	4
Reading a BIP39 phrase	6
See also	6

Bitcoin output script descriptors eliminate ambiguity when tracking, exporting, and reconstructing wallet scripts. Defined in BIP380, **key expressions** provide the exact cryptographic material required to construct script targets such as `pkh()`, `wpkh()`, and `tr()`. A descriptor key expression specifies not only the raw key string, but also optional origin metadata and derivation steps.

Understanding these expressions is essential when migrating wallets, building multisig setups, or auditing export files.

Structure of a BIP380 Key Expression

A key expression in BIP380 consists of up to three components: an optional key origin prefix, the key material itself, and an optional series of derivation steps.

The formal structure follows this syntax:

`[fingerprint/path]KEY/path/*`

- **Key origin info:** Enclosed in square brackets `[ . . . ]`. It begins with an 8-character hex **fingerprint** representing the first 4 bytes of the Hash160 of the base public key from which derivation began (typically the master key). This is followed by the slash-separated BIP32 derivation path that produced the key (such as hardened account path steps). The origin enables hardware signers to identify whether an imported descriptor belongs to their internal seed.
- **Base key:** The base key can be a fixed raw public key in hexadecimal, a private key in Wallet Import Format (**WIF**), an extended public key (`xpub`), or an extended private key (`xprv`).
- **Derivation steps:** Applied only to extended keys. Path elements can be fixed numerical indices or wildcards like `/*` or `/*'` (hardened). Multipath descriptors (standardized in BIP389) also support bracketed notation with semicolon-separated branch indices before a wildcard step, representing both external receive and internal change chains inside a single descriptor string.

Supported Key Formats

BIP380 standardizes four distinct base key types:

- 1 **Hex-encoded public keys:** Compressed public keys consist of 33 bytes (66 hexadecimal characters) starting with byte `02` or `03`. Uncompressed public keys consist of 65 bytes (130 hexadecimal characters) starting with byte `04`.
- 2 **WIF private keys:** Base58Check-encoded private keys. Uncompressed WIF keys begin with prefix `5` on mainnet (51 Base58 characters). Compressed WIF keys begin with prefix `K` or `L` on mainnet (52 Base58 characters). In BIP380, a WIF key represents an individual signing key for a specific script without child derivation.
- 3 **Extended public keys (`xpub`):** Serialized 78-byte BIP32 public structures encoded in Base58Check. They contain the chain code, depth, parent fingerprint, child index, and compressed public key, enabling unhardened derivation.
- 4 **Extended private keys (`xprv`):** Serialized 78-byte BIP32 private structures encoded in Base58Check. They contain full private signing data alongside chain code, enabling hardened and unhardened child key derivation.

Script Contexts and Compatibility Constraints

Not every key expression can be placed into every descriptor function. Different Bitcoin script contexts enforce strict cryptographic rules regarding compression and key length.

Script Function	Allowed Key Types	Compression Requirement	Derivation Support
<code>pk()</code> , <code>pkh()</code>	Hex pubkey, WIF, xpub, xprv	Compressed or uncompressed	Fixed keys, or BIP32 paths
<code>sh()</code>	Depends on wrapped script	Matches inner script requirements	Fixed keys, or BIP32 paths
<code>wpkh()</code> , <code>wsh()</code>	Hex pubkey, WIF, xpub, xprv	Compressed only (33 bytes)	Fixed keys, or BIP32 paths
<code>tr()</code>	Hex pubkey, WIF, xpub, xprv	32-byte x-only (derived from 33-byte)	Fixed keys, or BIP32 paths

In legacy scripts (`pkh()`), Bitcoin allows both compressed (33-byte) and uncompressed (65-byte) public keys. However, Segregated Witness v0 rules (`wpkh()` and `wsh()`) strictly require compressed public keys. Supplying an uncompressed hex key or an uncompressed WIF key to `wpkh()`, or inside a `wsh()` script, violates BIP141 standardness rules and is rejected by compliant descriptor parsers. Note that while `pk()`, `pkh()`, and `wpkh()` accept key expressions directly, `sh()` and `wsh()` accept script expressions whose inner scripts contain key expressions.

For Taproot (`tr()`, specified in BIP386), the consensus layer requires 32-byte x-only public keys as defined in BIP340. Inside a descriptor expression, users supply standard 33-byte compressed public keys (or xpub derivation paths). The `tr()` parser extracts the internal key by dropping the 1-byte prefix (ignoring the parity indicator) to yield the 32-byte x-coordinate automatically.

Private Keys in Descriptors and Security

A descriptor can serve as a watch-only template or as a complete spending wallet:

- **Public descriptors:** Contain only public keys (xpub or hex pubkeys). They derive receive and change addresses and monitor balances, but cannot sign transactions.
- **Private descriptors:** Contain private keys (xprv or WIF). These provide full spending authority over all derived outputs.

When a descriptor includes an xprv, it possesses the exact cryptographic capability of a BIP39 seed phrase for that account. Anyone with access to the descriptor string can derive all child private keys and drain the funds. Therefore, text backups of descriptors containing xprv or WIF strings must be treated with the same physical and cryptographic protections as a BIP39 recovery phrase.

Furthermore, never expose an xpub alongside any child private key from that same account. Because unhardened derivation allows deriving child keys via standard arithmetic, exposing an unhardened child private key along with the parent xpub permits an attacker to mathematically recover the parent private key.

FAQ

Can I derive hardened child keys from an xpub in a descriptor?

No. Hardened child derivation (`/*'` or `/*h`) mathematically requires the parent private key. An xpub base key in a descriptor can only derive unhardened child indices (`/*`). However, hardened derivation steps may appear in the key origin prefix preceding the xpub to indicate the path from the master key to that xpub.

What does the checksum at the end of a descriptor string mean?

A descriptor checksum consists of `#` followed by 8 characters using a BCH code. It is designed to catch most errors, including common typos, character swaps, and accidental truncation during copy-paste operations, though no checksum detects every possible combination of multiple arbitrary errors.

Why does `tr()` accept 33-byte keys if Taproot uses 32-byte keys?

BIP386 specifies `tr()` descriptors to accept standard 33-byte compressed keys or xpub derivations so that existing BIP32 derivation paths and key management tooling remain fully compatible. The parser discards the 1-byte parity prefix to yield the 32-byte BIP340 x-only internal key.

Sources

- [BIP 380: Output Script Descriptors General Operation \(github.com\)](#)
- [BIP 381: Non-Segwit Output Script Descriptors \(github.com\)](#)
- [BIP 386: `tr\(\)` Output Script Descriptors \(github.com\)](#)

Continue reading on bip39-phrase.com: [Bitcoin Private Key Format: WIF, Base58 and Hex Explained](#)
More on this topic: [BIP Standards](#)

BIP39 format examples

Randomly generated sequences of words from the BIP39 English wordlist. None of them passes the BIP39 checksum, so no wallet will accept them: they show the format only. Never store funds with a phrase you found in a document.

Example 1		
1	seed	2 swap
3	sheriff	
4	infant	5 crater
6	scale	
7	unit	8 assume
9	panic	
10	more	11 someone
12	clap	

Example 2

1 funny	2 galaxy	3 throw
4 alone	5 ceiling	6 apology
7 odor	8 mandate	9 boost
10 remind	11 whisper	12 fashion

Example 3

1 drop	2 axis	3 pipe
4 virus	5 tumble	6 apple
7 rocket	8 private	9 rebel
10 visual	11 dry	12 earn

Example 4

1 kid	2 west	3 genius
4 circle	5 kit	6 swap
7 human	8 main	9 mesh
10 fatal	11 panel	12 swim

Example 5

1 under	2 account	3 mimic
4 blur	5 thunder	6 then
7 task	8 biology	9 wait
10 athlete	11 age	12 column

Example 6

1 hazard	2 cube	3 antenna
4 census	5 smoke	6 model
7 grocery	8 away	9 spider
10 point	11 shiver	12 name

Example 7

1 govern	2 element	3 bounce
4 swear	5 wet	6 glue
7 estate	8 pool	9 wall
10 december	11 win	12 young

Example 8

1 midnight	2 dress	3 coach
4 detect	5 festival	6 index
7 index	8 shell	9 thank
10 solar	11 skull	12 erase

Reading a BIP39 phrase

Every word stands for an 11-bit number: its position in the 2048-word list, counted from 0. Here is Example 1 decoded word by word. Each word links to its page on bip39-phrased.com, where the list is numbered from 1.

#	Word	Index (0-2047)	Word no.	11 bits
1	seed	1559	1560	11000010111
2	swap	1754	1755	11011011010
3	sheriff	1580	1581	11000101100
4	infant	922	923	01110011010
5	crater	404	405	00110010100
6	scale	1536	1537	11000000000
7	unit	1901	1902	11101101101
8	assume	111	112	00001101111
9	panic	1277	1278	10011111101
10	more	1150	1151	10001111110
11	someone	1656	1657	11001111000
12	clap	334	335	00101001110

The 12 words carry 132 bits: 128 bits of entropy followed by a 4-bit checksum. The checksum must equal the first 4 bits of the SHA-256 hash of the entropy.

Entropy (hex): c2fb6b1639a329803b686f9fb1fb3c14

This sequence ends in the checksum bits 1110, but SHA-256 of its entropy starts with 0100, so wallets reject it as an invalid phrase. That is the point of the checksum: about 15 of every 16 random or mistyped sequences fail it.

See also

- [Watch-Only Wallets: Tracking Funds Without Private Keys](#) · October 18, 2026
- [Bitcoin Private Keys: Range, WIF and Hex Formats](#) · March 23, 2025
- [Compiling Policies with rust-miniscript](#) · October 20, 2026
- [Collaborative Custody: The Service Key Model](#) · May 7, 2026

Educational document. The example sequences are random and fail the BIP39 checksum: they are not wallets and must never hold funds. Security notes: [bip39-phrased.com/security/](#)