
libwally-core Documentation

Release 0.8.8

Jon Griffiths

Feb 10, 2023

Contents:

1	Core Functions	1
2	Crypto Functions	7
3	Address Functions	19
4	Bip32 Functions	25
5	Bip38 Functions	31
6	Bip39 Functions	33
7	Map Functions	37
8	Psbtx Functions	49
9	Script Functions	91
10	Descriptor Functions	101
11	Symmetric Functions	107
12	Transaction Functions	109
13	Elements Functions	131
14	Anti-Exfil Functions	143
15	Library Conventions	147
15.1	Error Codes	147
15.2	Variable Length Output Buffers	147
16	Liquid	149
16.1	Generating a confidential address	149
16.2	Receiving confidential assets	150
16.3	Spending confidential assets	151
17	Anti-Exfil Protocol	155
17.1	Step 1	155

17.2	Step 2	155
17.3	Step 3	156
17.4	Step 4	156
17.5	Step 5	156
18	Indices and tables	157
	Index	159

CHAPTER 1

Core Functions

int **wally_init** (uint32_t *flags*)

Initialize wally.

This function must be called once before threads are created by the application.

Parameters

- **flags** – Flags controlling what to initialize. Currently must be zero.

Returns See *Error Codes*

int **wally_cleanup** (uint32_t *flags*)

Free any internally allocated memory.

Parameters

- **flags** – Flags controlling what to clean up. Currently must be zero.

Returns See *Error Codes*

struct secp256k1_context_struct ***wally_get_secp_context** (void)

Fetch the wally internal secp256k1 context object.

By default, a single global context is created on demand. This behaviour can be overridden by providing a custom context fetching function when calling *wally_set_operations*.

struct secp256k1_context_struct ***wally_get_new_secp_context** (void)

Create a new wally-suitable secp256k1 context object.

The created context is initialised to be usable by all wally functions.

void **wally_secp_context_free** (struct secp256k1_context_struct **ctx*)

Free a secp256k1 context object created by *wally_get_new_secp_context*.

This function must only be called on context objects returned from *wally_get_new_secp_context*, it should not be called on the default context returned from *wally_get_secp_context*.

int **wally_bzero** (void **bytes*, size_t *bytes_len*)

Securely wipe memory.

Parameters

- **bytes** – Memory to wipe
- **bytes_len** – Size of **bytes** in bytes.

Returns See *Error Codes***int wally_free_string** (char **str*)

Securely wipe and then free a string allocated by the library.

Parameters

- **str** – String to free (must be NUL terminated UTF-8).

Returns See *Error Codes***int wally_secp_randomize** (const unsigned char **bytes*, size_t *bytes_len*)

Provide entropy to randomize the libraries internal libsecp256k1 context.

Random data is used in libsecp256k1 to blind the data being processed, making side channel attacks more difficult. By default, Wally uses a single internal context for secp functions that is not initially randomized.

The caller should call this function before using any functions that rely on libsecp256k1 (i.e. Anything using public/private keys). If the caller has overridden the library's default libsecp context fetching using *wally_set_operations*, then it may be necessary to call this function before calling wally functions in each thread created by the caller.

If wally is used in its default configuration, this function should either be called before threads are created or access to wally functions wrapped in an application level mutex.

Parameters

- **bytes** – Entropy to use.
- **bytes_len** – Size of **bytes** in bytes. Must be WALLY_SECP_RANDOMIZE_LEN.

Returns See *Error Codes***int wally_hex_verify** (const char **hex*)

Verify that a hexadecimal string is valid.

Parameters

- **hex** – String to verify.

Returns See *Error Codes***int wally_hex_n_verify** (const char **hex*, size_t *hex_len*)

Verify that a known-length hexadecimal string is valid.

See *wally_hex_verify*.**Returns** See *Error Codes***int wally_hex_from_bytes** (const unsigned char **bytes*, size_t *bytes_len*, char ***output*)

Convert bytes to a (lower-case) hexadecimal string.

Parameters

- **bytes** – Bytes to convert.
- **bytes_len** – Size of **bytes** in bytes.
- **output** – Destination for the resulting hexadecimal string. The string returned should be freed using *wally_free_string*.

Returns See *Error Codes*

int **wally_hex_to_bytes** (const char *hex, unsigned char *bytes_out, size_t len, size_t *written)
Convert a hexadecimal string to bytes.

Parameters

- **hex** – String to convert.
- **bytes_out** – Where to store the resulting bytes.
- **len** – The length of bytes_out in bytes.
- **written** – Destination for the number of bytes written to bytes_out.

Returns See *Variable Length Output Buffers*

int **wally_hex_n_to_bytes** (const char *hex, size_t hex_len, unsigned char *bytes_out, size_t len, size_t *written)
Convert a known-length hexadecimal string to bytes.

See *wally_hex_to_bytes*.

Returns See *Variable Length Output Buffers*

int **wally_base58_from_bytes** (const unsigned char *bytes, size_t bytes_len, uint32_t flags, char **out-put)
Create a base 58 encoded string representing binary data.

Parameters

- **bytes** – Binary data to convert.
- **bytes_len** – The length of bytes in bytes.
- **flags** – Pass BASE58_FLAG_CHECKSUM if bytes should have a checksum calculated and appended before converting to base 58.
- **output** – Destination for the base 58 encoded string representing bytes. The string returned should be freed using *wally_free_string*.

Returns See *Error Codes*

int **wally_base58_to_bytes** (const char *str_in, uint32_t flags, unsigned char *bytes_out, size_t len, size_t *written)
Decode a base 58 encoded string back into binary data.

Parameters

- **str_in** – Base 58 encoded string to decode.
- **flags** – Pass BASE58_FLAG_CHECKSUM if bytes_out should have a checksum validated and removed before returning. In this case, len must contain an extra BASE58_CHECKSUM_LEN bytes to calculate the checksum into. The returned length will not include the checksum.
- **bytes_out** – Destination for converted binary data.
- **len** – The length of bytes_out in bytes.
- **written** – Destination for the length of the decoded bytes.

Returns See *Variable Length Output Buffers*

int **wally_base58_n_to_bytes** (const char *str_in, size_t str_len, uint32_t flags, unsigned char *bytes_out, size_t len, size_t *written)
Decode a known-length base 58 encoded string back into binary data.

See *wally_base58_to_bytes*.

Returns See *Variable Length Output Buffers*

int **wally_base58_get_length** (const char **str_in*, size_t **written*)

Return the length of a base 58 encoded string once decoded into bytes.

Returns the exact number of bytes that would be required to store *str_in* as decoded binary, including any embedded checksum. If the string contains invalid characters then WALLY_EINVAL is returned. Note that no checksum validation takes place.

In the worst case (an all zero buffer, represented by a string of '1' characters), this function will return `strlen(str_in)`. You can therefore safely use the length of *str_in* as a buffer size to avoid calling this function in most cases.

Parameters

- **str_in** – Base 58 encoded string to find the length of.
- **written** – Destination for the length of the decoded bytes.

Returns See *Error Codes*

int **wally_base58_n_get_length** (const char **str_in*, size_t *str_len*, size_t **written*)

Return the length of a known-length base 58 encoded string once decoded into bytes.

See *wally_base58_get_length*.

Returns See *Error Codes*

int **wally_base64_from_bytes** (const unsigned char **bytes*, size_t *bytes_len*, uint32_t *flags*, char ***output*)

Create a base64 encoded string representing binary data.

Parameters

- **bytes** – Binary data to convert.
- **bytes_len** – The length of *bytes* in bytes.
- **flags** – Must be 0.
- **output** – Destination for the base64 encoded string representing bytes. The string returned should be freed using *wally_free_string*.

Returns See *Error Codes*

int **wally_base64_to_bytes** (const char **str_in*, uint32_t *flags*, unsigned char **bytes_out*, size_t *len*, size_t **written*)

Decode a base64 encoded string back into binary data.

Parameters

- **str_in** – Base64 encoded string to decode.
- **flags** – Must be 0.
- **bytes_out** – Destination for converted binary data.
- **len** – The length of *bytes_out* in bytes. See *wally_base64_get_maximum_length*.
- **written** – Destination for the length of the decoded bytes.

Returns See *Variable Length Output Buffers*

int **wally_base64_get_maximum_length** (const char **str_in*, uint32_t *flags*, size_t **written*)

Return the maximum length of a base64 encoded string once decoded into bytes.

Since base64 strings may contain line breaks and padding, it is not possible to compute their decoded length without fully decoding them. This function cheaply calculates the maximum possible decoded length, which can be used to allocate a buffer for `wally_base64_to_bytes`. In most cases the decoded data will be shorter than the value returned.

Parameters

- **str_in** – Base64 encoded string to find the length of.
- **flags** – Must be 0.
- **written** – Destination for the maximum length of the decoded bytes.

Returns See [Error Codes](#)

int **wally_get_operations** (struct wally_operations *output)

Fetch the current overridable operations used by wally.

Parameters

- **output** – Destination for the overridable operations.

Returns See [Error Codes](#)

int **wally_set_operations** (const struct wally_operations *ops)

Set the current overridable operations used by wally.

Parameters

- **ops** – The overridable operations to set.

Note: Any NULL members in the passed structure are ignored.

Returns See [Error Codes](#)

int **wally_is_elements_build** (size_t *written)

Determine if the library was built with elements support.

Parameters

- **written** – 1 if the library supports elements, otherwise 0.

Returns See [Error Codes](#)

Crypto Functions

int **wally_script** (const unsigned char **pass*, size_t *pass_len*, const unsigned char **salt*, size_t *salt_len*,
uint32_t *cost*, uint32_t *block_size*, uint32_t *parallelism*, unsigned char **bytes_out*,
size_t *len*)

Derive a pseudorandom key from inputs using an expensive application of HMAC SHA-256.

Parameters

- **pass** – Password to derive from.
- **pass_len** – Length of *pass* in bytes.
- **salt** – Salt to derive from.
- **salt_len** – Length of *salt* in bytes.
- **cost** – The cost of the function. The larger this number, the longer the key will take to derive.
- **block_size** – The size of memory blocks required.
- **parallelism** – Parallelism factor.
- **bytes_out** – Destination for the derived pseudorandom key.
- **len** – The length of *bytes_out* in bytes. Must be a non-zero multiple of PBKDF2_HMAC_SHA256_LEN.

Returns See [Error Codes](#)

int **wally_aes** (const unsigned char **key*, size_t *key_len*, const unsigned char **bytes*, size_t *bytes_len*,
uint32_t *flags*, unsigned char **bytes_out*, size_t *len*)

Encrypt/decrypt data using AES (ECB mode, no padding).

Parameters

- **key** – Key material for initialisation.
- **key_len** – Length of *key* in bytes. Must be an **AES_KEY_LEN** constant.
- **bytes** – Bytes to encrypt/decrypt.

- **bytes_len** – Length of **bytes** in bytes. Must be a multiple of **AES_BLOCK_LEN**.
- **flags** – **AES_FLAG_** constants indicating the desired behavior.
- **bytes_out** – Destination for the encrypted/decrypted data.
- **len** – The length of **bytes_out** in bytes. Must be a multiple of **AES_BLOCK_LEN**.

Returns See [Error Codes](#)

int **wally_aes_cbc** (const unsigned char *key, size_t key_len, const unsigned char *iv, size_t iv_len, const unsigned char *bytes, size_t bytes_len, uint32_t flags, unsigned char *bytes_out, size_t len, size_t *written)
Encrypt/decrypt data using AES (CBC mode, PKCS#7 padding).

Parameters

- **key** – Key material for initialisation.
- **key_len** – Length of **key** in bytes. Must be an **AES_KEY_LEN** constant.
- **iv** – Initialisation vector.
- **iv_len** – Length of **iv** in bytes. Must be **AES_BLOCK_LEN**.
- **bytes** – Bytes to encrypt/decrypt.
- **bytes_len** – Length of **bytes** in bytes. Can be of any length for encryption, must be a multiple of **AES_BLOCK_LEN** for decryption.
- **flags** – **AES_FLAG_** constants indicating the desired behavior.
- **bytes_out** – Destination for the encrypted/decrypted data.
- **len** – The length of **bytes_out** in bytes. Must be a multiple of **AES_BLOCK_LEN**.
- **written** – Destination for the number of bytes written to **bytes_out**.

Returns See [Variable Length Output Buffers](#)

int **wally_sha256** (const unsigned char *bytes, size_t bytes_len, unsigned char *bytes_out, size_t len)
SHA-256(m)

Parameters

- **bytes** – The message to hash.
- **bytes_len** – The length of **bytes** in bytes.
- **bytes_out** – Destination for the resulting hash.
- **len** – Size of **bytes_out**. Must be **SHA256_LEN**.

Returns See [Error Codes](#)

int **wally_sha256_midstate** (const unsigned char *bytes, size_t bytes_len, unsigned char *bytes_out, size_t len)
SHA-256(m) midstate

Parameters

- **bytes** – The message to hash.
- **bytes_len** – The length of **bytes** in bytes.
- **bytes_out** – Destination for the resulting hash.
- **len** – Size of **bytes_out**. Must be **SHA256_LEN**.

Returns See [Error Codes](#)

int **wally_sha256d** (const unsigned char *bytes, size_t bytes_len, unsigned char *bytes_out, size_t len)
SHA-256(SHA-256(m)) (double SHA-256).

Parameters

- **bytes** – The message to hash.
- **bytes_len** – The length of bytes in bytes.
- **bytes_out** – Destination for the resulting hash.
- **len** – Size of bytes_out. Must be SHA256_LEN.

Returns See [Error Codes](#)

int **wally_sha512** (const unsigned char *bytes, size_t bytes_len, unsigned char *bytes_out, size_t len)
SHA-512(m).

Parameters

- **bytes** – The message to hash.
- **bytes_len** – The length of bytes in bytes.
- **bytes_out** – Destination for the resulting hash.
- **len** – Size of bytes_out. Must be SHA512_LEN.

Returns See [Error Codes](#)

int **wally_bip340_tagged_hash** (const unsigned char *bytes, size_t bytes_len, const char *tag, unsigned char *bytes_out, size_t len)
BIP340 tagged hash: SHA-256(SHA-256(tag) || SHA-256(tag) || m).

Parameters

- **bytes** – The message to hash.
- **bytes_len** – The length of bytes in bytes.
- **tag** – The BIP340 UTF-8 domain tag.
- **bytes_out** – Destination for the resulting hash.
- **len** – Size of bytes_out. Must be SHA256_LEN.

Returns See [Error Codes](#)

int **wally_ripemd160** (const unsigned char *bytes, size_t bytes_len, unsigned char *bytes_out, size_t len)
RIPEMD-160(m).

Parameters

- **bytes** – The message to hash.
- **bytes_len** – The length of bytes in bytes.
- **bytes_out** – Destination for the resulting hash.
- **len** – Size of bytes_out. Must be RIPEMD160_LEN.

Returns See [Error Codes](#)

int **wally_hash160** (const unsigned char *bytes, size_t bytes_len, unsigned char *bytes_out, size_t len)
RIPEMD-160(SHA-256(m)).

Parameters

- **bytes** – The message to hash.

- **bytes_len** – The length of `bytes` in bytes.
- **bytes_out** – Destination for the resulting hash.
- **len** – Size of `bytes_out`. Must be `HASH160_LEN`.

Returns See *Error Codes*

int **wally_hmac_sha256**(const unsigned char *key, size_t key_len, const unsigned char *bytes,
size_t bytes_len, unsigned char *bytes_out, size_t len)
Compute an HMAC using SHA-256.

Parameters

- **key** – The key for the hash.
- **key_len** – The length of `key` in bytes.
- **bytes** – The message to hash.
- **bytes_len** – The length of `bytes` in bytes.
- **bytes_out** – Destination for the resulting HMAC.
- **len** – Size of `bytes_out`. Must be `HMAC_SHA256_LEN`.

Returns See *Error Codes*

int **wally_hmac_sha512**(const unsigned char *key, size_t key_len, const unsigned char *bytes,
size_t bytes_len, unsigned char *bytes_out, size_t len)
Compute an HMAC using SHA-512.

Parameters

- **key** – The key for the hash.
- **key_len** – The length of `key` in bytes.
- **bytes** – The message to hash.
- **bytes_len** – The length of `bytes` in bytes.
- **bytes_out** – Destination for the resulting HMAC.
- **len** – Size of `bytes_out`. Must be `HMAC_SHA512_LEN`.

Returns See *Error Codes*

int **wally_pbkdf2_hmac_sha256**(const unsigned char *pass, size_t pass_len, const unsigned char *salt,
size_t salt_len, uint32_t flags, uint32_t cost, unsigned char *bytes_out,
size_t len)
Derive a pseudorandom key from inputs using HMAC SHA-256.

Parameters

- **pass** – Password to derive from.
- **pass_len** – Length of `pass` in bytes.
- **salt** – Salt to derive from.
- **salt_len** – Length of `salt` in bytes.
- **flags** – Reserved, must be 0.
- **cost** – The cost of the function. The larger this number, the longer the key will take to derive.
- **bytes_out** – Destination for the derived pseudorandom key.

- **len** – Size of `bytes_out`. Must be `PBKDF2_HMAC_SHA256_LEN`.

Returns See *Error Codes*

int **wally_pbkdf2_hmac_sha512** (const unsigned char **pass*, size_t *pass_len*, const unsigned char **salt*, size_t *salt_len*, uint32_t *flags*, uint32_t *cost*, unsigned char **bytes_out*, size_t *len*)

Derive a pseudorandom key from inputs using HMAC SHA-512.

Parameters

- **pass** – Password to derive from.
- **pass_len** – Length of `pass` in bytes.
- **salt** – Salt to derive from.
- **salt_len** – Length of `salt` in bytes.
- **flags** – Reserved, must be 0.
- **cost** – The cost of the function. The larger this number, the longer the key will take to derive.
- **bytes_out** – Destination for the derived pseudorandom key.
- **len** – Size of `bytes_out`. Must be `PBKDF2_HMAC_SHA512_LEN`.

Returns See *Error Codes*

int **wally_ec_private_key_verify** (const unsigned char **priv_key*, size_t *priv_key_len*)

Verify that a private key is valid.

Parameters

- **priv_key** – The private key to validate.
- **priv_key_len** – The length of `priv_key` in bytes. Must be `EC_PRIVATE_KEY_LEN`.

Returns See *Error Codes*

int **wally_ec_public_key_verify** (const unsigned char **pub_key*, size_t *pub_key_len*)

Verify that a public key is valid.

Parameters

- **pub_key** – The public key to validate.
- **pub_key_len** – The length of `pub_key` in bytes. Must be `EC_PUBLIC_KEY_LEN` or `EC_PUBLIC_KEY_UNCOMPRESSED_LEN`.

Returns See *Error Codes*

int **wally_ec_xonly_public_key_verify** (const unsigned char **pub_key*, size_t *pub_key_len*)

Verify that an x-only public key is valid.

Parameters

- **pub_key** – The x-only public key to validate.
- **pub_key_len** – The length of `pub_key` in bytes. Must be `EC_XONLY_PUBLIC_KEY_LEN`.

Returns See *Error Codes*

int **wally_ec_public_key_from_private_key** (const unsigned char *priv_key, size_t priv_key_len,
unsigned char *bytes_out, size_t len)

Create a public key from a private key.

Parameters

- **priv_key** – The private key to create a public key from.
- **priv_key_len** – The length of priv_key in bytes. Must be EC_PRIVATE_KEY_LEN.
- **bytes_out** – Destination for the resulting public key.
- **len** – Size of bytes_out. Must be EC_PUBLIC_KEY_LEN.

Returns See *Error Codes*

int **wally_ec_public_key_decompress** (const unsigned char *pub_key, size_t pub_key_len, unsigned
char *bytes_out, size_t len)

Create an uncompressed public key from a compressed public key.

Parameters

- **pub_key** – The public key to decompress.
- **pub_key_len** – The length of pub_key in bytes. Must be EC_PUBLIC_KEY_LEN.
- **bytes_out** – Destination for the resulting public key.
- **len** – Size of bytes_out. Must be EC_PUBLIC_KEY_UNCOMPRESSED_LEN.

Returns See *Error Codes*

int **wally_ec_public_key_negate** (const unsigned char *pub_key, size_t pub_key_len, unsigned
char *bytes_out, size_t len)

Negates a public key.

Parameters

- **pub_key** – The public key to negate.
- **pub_key_len** – The length of pub_key in bytes. Must be EC_PUBLIC_KEY_LEN.
- **bytes_out** – Destination for the resulting public key.
- **len** – Size of bytes_out. Must be EC_PUBLIC_KEY_LEN.

Returns See *Error Codes*

int **wally_ec_sig_from_bytes_len** (const unsigned char *priv_key, size_t priv_key_len, const un-
signed char *bytes, size_t bytes_len, uint32_t flags, size_t *writ-
ten)

Get the expected length of a signature in bytes.

Parameters

- **priv_key** – The private key to sign with.
- **priv_key_len** – The length of priv_key in bytes. Must be EC_PRIVATE_KEY_LEN.
- **bytes** – The message hash to sign.
- **bytes_len** – The length of bytes in bytes. Must be EC_MESSAGE_HASH_LEN.
- **flags** – **EC_FLAG_** flag values indicating desired behavior.
- **written** – Destination for the expected length of the signature, either EC_SIGNATURE_LEN or EC_SIGNATURE_RECOVERABLE_LEN.

Returns See *Error Codes*

int **wally_ec_sig_from_bytes** (const unsigned char *priv_key, size_t priv_key_len, const unsigned char *bytes, size_t bytes_len, uint32_t flags, unsigned char *bytes_out, size_t len)

Sign a message hash with a private key, producing a compact signature.

Parameters

- **priv_key** – The private key to sign with.
- **priv_key_len** – The length of priv_key in bytes. Must be EC_PRIVATE_KEY_LEN.
- **bytes** – The message hash to sign.
- **bytes_len** – The length of bytes in bytes. Must be EC_MESSAGE_HASH_LEN.
- **flags** – **EC_FLAG_** flag values indicating desired behavior.
- **bytes_out** – Destination for the resulting compact signature.
- **len** – The length of bytes_out in bytes. Must be EC_SIGNATURE_RECOVERABLE_LEN if flags includes EC_FLAG_RECOVERABLE, otherwise EC_SIGNATURE_LEN.

Returns See *Error Codes*

int **wally_ec_sig_normalize** (const unsigned char *sig, size_t sig_len, unsigned char *bytes_out, size_t len)

Convert a signature to low-s form.

Parameters

- **sig** – The compact signature to convert.
- **sig_len** – The length of sig in bytes. Must be EC_SIGNATURE_LEN.
- **bytes_out** – Destination for the resulting low-s signature.
- **len** – Size of bytes_out. Must be EC_SIGNATURE_LEN.

Returns See *Error Codes*

int **wally_ec_sig_to_der** (const unsigned char *sig, size_t sig_len, unsigned char *bytes_out, size_t len, size_t *written)

Convert a compact signature to DER encoding.

Parameters

- **sig** – The compact signature to convert.
- **sig_len** – The length of sig in bytes. Must be EC_SIGNATURE_LEN.
- **bytes_out** – Destination for the resulting DER encoded signature.
- **n** – Size of bytes_out. Passing EC_SIGNATURE_DER_MAX_LEN will ensure the buffer is large enough.
- **written** – Destination for the number of bytes written to bytes_out.

Returns See *Variable Length Output Buffers*

int **wally_ec_sig_from_der** (const unsigned char *bytes, size_t bytes_len, unsigned char *bytes_out, size_t len)

Convert a DER encoded signature to a compact signature.

Parameters

- **bytes** – The DER encoded signature to convert.
- **bytes_len** – The length of `sig` in bytes.
- **bytes_out** – Destination for the resulting compact signature.
- **len** – Size of `bytes_out`. Must be `EC_SIGNATURE_LEN`.

Returns See *Error Codes*

int **wally_ec_sig_verify** (const unsigned char **pub_key*, size_t *pub_key_len*, const unsigned char **bytes*, size_t *bytes_len*, uint32_t *flags*, const unsigned char **sig*, size_t *sig_len*)
Verify a signed message hash.

Parameters

- **pub_key** – The public key to verify with.
- **pub_key_len** – The length of `pub_key` in bytes. Must be `EC_PUBLIC_KEY_LEN`.
- **bytes** – The message hash to verify.
- **bytes_len** – The length of `bytes` in bytes. Must be `EC_MESSAGE_HASH_LEN`.
- **flags** – **EC_FLAG** flag values indicating desired behavior.
- **sig** – The compact signature of the message in bytes.
- **sig_len** – The length of `sig` in bytes. Must be `EC_SIGNATURE_LEN`.

Returns See *Error Codes*

int **wally_ec_sig_to_public_key** (const unsigned char **bytes*, size_t *bytes_len*, const unsigned char **sig*, size_t *sig_len*, unsigned char **bytes_out*, size_t *len*)
Recover compressed public key from a recoverable signature.

Parameters

- **bytes** – The message hash signed.
- **bytes_len** – The length of `bytes` in bytes. Must be `EC_MESSAGE_HASH_LEN`.
- **sig** – The recoverable compact signature of the message in bytes.
- **sig_len** – The length of `sig` in bytes. Must be `EC_SIGNATURE_RECOVERABLE_LEN`.
- **bytes_out** – Destination for recovered public key.
- **len** – Size of `bytes_out`. Must be `EC_PUBLIC_KEY_LEN`.

Note: The successful recovery of the public key guarantees the correctness of the signature.

Returns See *Error Codes*

int **wally_ec_scalar_verify** (const unsigned char **scalar*, size_t *scalar_len*)
Verify that a secp256k1 scalar value is valid.

Parameters

- **scalar** – The starting scalar to have a value added to.
- **scalar_len** – The length of `scalar` in bytes. Must be `EC_SCALAR_LEN`.

Returns See *Error Codes*

```
int wally_ec_scalar_add (const unsigned char *scalar, size_t scalar_len, const unsigned char *operand,  
                        size_t operand_len, unsigned char *bytes_out, size_t len)
```

Add one secp256k1 scalar to another.

Parameters

- **scalar** – The starting scalar to have a value added to.
- **scalar_len** – The length of scalar in bytes. Must be EC_SCALAR_LEN.
- **operand** – The scalar value to add to scalar.
- **operand_len** – The length of operand in bytes. Must be EC_SCALAR_LEN.
- **bytes_out** – Destination for the resulting scalar.
- **len** – Size of bytes_out. Must be EC_SCALAR_LEN.

Note: Computes (scalar + operand) % n. Returns WALLY_ERROR if either input is not within the secp256k1 group order n.

Returns See [Error Codes](#)

```
int wally_ec_scalar_subtract (const unsigned char *scalar, size_t scalar_len, const unsigned  
                             char *operand, size_t operand_len, unsigned char *bytes_out,  
                             size_t len)
```

Subtract one secp256k1 scalar from another.

Parameters

- **scalar** – The starting scalar to have a value subtracted from.
- **scalar_len** – The length of scalar in bytes. Must be EC_SCALAR_LEN.
- **operand** – The scalar value to subtract from scalar.
- **operand_len** – The length of operand in bytes. Must be EC_SCALAR_LEN.
- **bytes_out** – Destination for the resulting scalar.
- **len** – Size of bytes_out. Must be EC_SCALAR_LEN.

Note: Computes (scalar - operand) % n. Returns WALLY_ERROR if either input is not within the secp256k1 group order n.

Returns See [Error Codes](#)

```
int wally_ec_scalar_multiply (const unsigned char *scalar, size_t scalar_len, const unsigned  
                             char *operand, size_t operand_len, unsigned char *bytes_out,  
                             size_t len)
```

Multiply one secp256k1 scalar by another.

Parameters

- **scalar** – The starting scalar to multiply.
- **scalar_len** – The length of scalar in bytes. Must be EC_SCALAR_LEN.
- **operand** – The scalar value to multiply scalar by.
- **operand_len** – The length of operand in bytes. Must be EC_SCALAR_LEN.

- **bytes_out** – Destination for the resulting scalar.
- **len** – Size of bytes_out. Must be EC_SCALAR_LEN.

Note: Computes (scalar * operand) % n. Returns WALLY_ERROR if either input is not within the secp256k1 group order n.

Returns See *Error Codes*

int **wally_ec_scalar_add_to**(unsigned char *scalar, size_t scalar_len, const unsigned char *operand, size_t operand_len)
Add one secp256k1 scalar to another in place.

Note: As per *wally_ec_scalar_add* with scalar modified in place.

Returns See *Error Codes*

int **wally_ec_scalar_subtract_from**(unsigned char *scalar, size_t scalar_len, const unsigned char *operand, size_t operand_len)
Subtract one secp256k1 scalar from another in place.

Note: As per *wally_ec_scalar_subtract* with scalar modified in place.

Returns See *Error Codes*

int **wally_ec_scalar_multiply_by**(unsigned char *scalar, size_t scalar_len, const unsigned char *operand, size_t operand_len)
Multiply one secp256k1 scalar by another in place.

Note: As per *wally_ec_scalar_multiply* with scalar modified in place.

Returns See *Error Codes*

int **wally_format_bitcoin_message**(const unsigned char *bytes, size_t bytes_len, uint32_t flags, unsigned char *bytes_out, size_t len, size_t *written)
Format a message for use as a bitcoin signed message.

Parameters

- **bytes** – The message string to sign.
- **bytes_len** – The length of bytes in bytes. Must be less than or equal to BITCOIN_MESSAGE_MAX_LEN.
- **flags** – **BITCOIN_MESSAGE_FLAG** flags indicating the desired output. if BITCOIN_MESSAGE_FLAG_HASH is passed, the double SHA256 hash of the message is placed in bytes_out instead of the formatted message. In this case len must be at least SHA256_LEN.
- **bytes_out** – Destination for the formatted message or message hash.
- **len** – The length of bytes_out in bytes.

- **written** – Destination for the number of bytes written to `bytes_out`.

Returns See *Variable Length Output Buffers*

int **wally_ecdh** (const unsigned char **pub_key*, size_t *pub_key_len*, const unsigned char **priv_key*, size_t *priv_key_len*, unsigned char **bytes_out*, size_t *len*)

Compute an EC Diffie-Hellman secret in constant time.

Parameters

- **pub_key** – The public key.
- **pub_key_len** – The length of `pub_key` in bytes. Must be `EC_PUBLIC_KEY_LEN`.
- **priv_key** – The private key.
- **priv_key_len** – The length of `priv_key` in bytes. Must be `EC_PRIVATE_KEY_LEN`.
- **bytes_out** – Destination for the shared secret.
- **len** – Size of `bytes_out`. Must be `SHA256_LEN`.

Note: If `priv_key` is invalid, this call returns `WALLY_ERROR`.

Returns See *Error Codes*

int **wally_s2c_sig_from_bytes** (const unsigned char **priv_key*, size_t *priv_key_len*, const unsigned char **bytes*, size_t *bytes_len*, const unsigned char **s2c_data*, size_t *s2c_data_len*, uint32_t *flags*, unsigned char **s2c_opening_out*, size_t *s2c_opening_out_len*, unsigned char **bytes_out*, size_t *len*)

Sign a message hash with a private key, producing a compact signature which commits to additional data using sign-to-contract (s2c).

Parameters

- **priv_key** – The private key to sign with.
- **priv_key_len** – The length of `priv_key` in bytes. Must be `EC_PRIVATE_KEY_LEN`.
- **bytes** – The message hash to sign.
- **bytes_len** – The length of `bytes` in bytes. Must be `EC_MESSAGE_HASH_LEN`.
- **s2c_data** – The data to commit to.
- **s2c_data_len** – The length of `s2c_data` in bytes. Must be `WALLY_S2C_DATA_LEN`.
- **flags** – Must be `EC_FLAG_ECDSA`.
- **s2c_opening_out** – Destination for the resulting opening information.
- **s2c_opening_out_len** – Size of `s2c_opening_out`. Must be `WALLY_S2C_OPENING_LEN`.
- **bytes_out** – Destination for the resulting compact signature.
- **len** – Size of `bytes_out`. Must be `EC_SIGNATURE_LEN`.

Returns See *Error Codes*

```
int wally_s2c_commitment_verify(const unsigned char *sig, size_t sig_len, const unsigned
                                char *s2c_data, size_t s2c_data_len, const unsigned
                                char *s2c_opening, size_t s2c_opening_len, uint32_t flags)
```

Verify a sign-to-contract (s2c) commitment.

Parameters

- **sig** – The compact signature.
- **sig_len** – The length of `sig` in bytes. Must be `EC_SIGNATURE_LEN`.
- **s2c_data** – The data that was committed to.
- **s2c_data_len** – The length of `s2c_data` in bytes. Must be `WALLY_S2C_DATA_LEN`.
- **s2c_opening** – The opening information produced during signing.
- **s2c_opening_len** – The length of `s2c_opening` in bytes. Must be `WALLY_S2C_OPENING_LEN`.
- **flags** – Must be `EC_FLAG_ECDSA`.

Returns See *Error Codes*

Address Functions

int **wally_addr_segwit_from_bytes** (const unsigned char *bytes, size_t bytes_len, const char *addr_family, uint32_t flags, char **output)

Create a segwit native address from a v0 or later witness program.

Parameters

- **bytes** – Witness program bytes, including the version and data push opcode.
- **bytes_len** – Length of bytes in bytes. Must be HASH160_LEN or SHA256_LEN for v0 witness programs.
- **addr_family** – Address family to generate, e.g. “bc” or “tb”.
- **flags** – For future use. Must be 0.
- **output** – Destination for the resulting segwit native address string.

Returns See *Error Codes*

int **wally_addr_segwit_to_bytes** (const char *addr, const char *addr_family, uint32_t flags, unsigned char *bytes_out, size_t len, size_t *written)

Get a scriptPubKey containing the witness program from a segwit native address.

Parameters

- **addr** – Address to fetch the witness program from.
- **addr_family** – Address family to generate, e.g. “bc” or “tb”.
- **flags** – For future use. Must be 0.
- **bytes_out** – Destination for the resulting scriptPubKey (including the version and data push opcode)
- **n** – Size of bytes_out. Passing WALLY_SEGWIT_ADDRESS_PUBKEY_MAX_LEN will ensure the buffer is large enough.
- **written** – Destination for the number of bytes written to bytes_out.

Returns See *Variable Length Output Buffers*

```
int wally_addr_segwit_n_to_bytes (const char *addr, size_t addr_len, const char *addr_family,
                                size_t addr_family_len, uint32_t flags, unsigned char *bytes_out,
                                size_t len, size_t *written)
```

Get a scriptPubKey containing the witness program from a known-length segwit native address.

See [wally_addr_segwit_to_bytes](#).

Parameters

- **n** – Size of `bytes_out`. Passing `WALLY_SEGWIT_ADDRESS_PUBKEY_MAX_LEN` will ensure the buffer is large enough.

Returns See [Variable Length Output Buffers](#)

```
int wally_addr_segwit_get_version (const char *addr, const char *addr_family, uint32_t flags,
                                size_t *written)
```

Get the segwit version of a segwit native address.

Parameters

- **addr** – Address to fetch the witness program from.
- **addr_family** – Address family to generate, e.g. “bc” or “tb”.
- **flags** – For future use. Must be 0.
- **written** – Destination for the segwit version from 0 to 16 inclusive.

Returns See [Error Codes](#)

```
int wally_addr_segwit_n_get_version (const char *addr, size_t addr_len, const char *addr_family,
                                size_t addr_family_len, uint32_t flags, size_t *written)
```

Get the segwit version of a known-length segwit native address.

See [wally_addr_segwit_get_version](#).

Returns See [Error Codes](#)

```
int wally_address_to_scriptpubkey (const char *addr, uint32_t network, unsigned char *bytes_out,
                                size_t len, size_t *written)
```

Infer a scriptPubKey from an address.

Parameters

- **addr** – Base58 encoded address to infer the scriptPubKey from. For confidential Liquid addresses first call [wally_confidential_addr_to_addr\(\)](#)
- **network** – One of `WALLY_NETWORK_BITCOIN_MAINNET`, `WALLY_NETWORK_BITCOIN_TESTNET`, `WALLY_NETWORK_LIQUID`, `WALLY_NETWORK_LIQUID_REGTEST`.
- **bytes_out** – Destination for the resulting scriptPubKey
- **n** – Size of `bytes_out`. Passing `WALLY_ADDRESS_PUBKEY_MAX_LEN` will ensure the buffer is large enough.
- **written** – Destination for the number of bytes written to `bytes_out`.

Returns See [Variable Length Output Buffers](#)

```
int wally_scriptpubkey_to_address (const unsigned char *scriptpubkey, size_t scriptpubkey_len,
                                uint32_t network, char **output)
```

Infer address from a scriptPubKey. For SegWit addresses, use [wally_addr_segwit_from_bytes](#) instead. To find out if an address is SegWit, use [wally_scriptpubkey_get_type](#).

Parameters

- **scriptpubkey** – scriptPubKey bytes.
- **scriptpubkey_len** – Length of scriptpubkey in bytes.
- **network** – One of WALLY_NETWORK_BITCOIN_MAINNET, WALLY_NETWORK_BITCOIN_TESTNET, WALLY_NETWORK_LIQUID, WALLY_NETWORK_LIQUID_REGTEST.
- **output** – Destination for the resulting Base58 encoded address string.

Returns See *Error Codes*

int **wally_wif_from_bytes** (const unsigned char *priv_key, size_t priv_key_len, uint32_t prefix, uint32_t flags, char **output)
Convert a private key to Wallet Import Format.

Parameters

- **priv_key** – Private key bytes.
- **priv_key_len** – The length of priv_key in bytes. Must be EC_PRIVATE_KEY_LEN.
- **prefix** – Expected prefix byte, e.g. WALLY_ADDRESS_VERSION_WIF_MAINNET or WALLY_ADDRESS_VERSION_WIF_TESTNET.
- **flags** – Pass WALLY_WIF_FLAG_COMPRESSED if the corresponding pubkey is compressed, otherwise WALLY_WIF_FLAG_UNCOMPRESSED.
- **output** – Destination for the resulting Wallet Import Format string.

Returns See *Error Codes*

int **wally_wif_to_bytes** (const char *wif, uint32_t prefix, uint32_t flags, unsigned char *bytes_out, size_t len)
Convert a Wallet Import Format string to a private key.

Parameters

- **wif** – Private key in Wallet Import Format.
- **prefix** – Prefix byte to use, e.g. WALLY_ADDRESS_VERSION_WIF_MAINNET or WALLY_ADDRESS_VERSION_WIF_TESTNET.
- **flags** – Pass WALLY_WIF_FLAG_COMPRESSED if the corresponding pubkey is compressed, otherwise WALLY_WIF_FLAG_UNCOMPRESSED.
- **bytes_out** – Destination for the private key.
- **len** – Size of bytes_out. Must be EC_PRIVATE_KEY_LEN.

Returns See *Error Codes*

int **wally_wif_is_uncompressed** (const char *wif, size_t *written)
Determine if a private key in Wallet Import Format corresponds to an uncompressed public key.

Parameters

- **wif** – Private key in Wallet Import Format to check.
- **written** – 1 if the corresponding public key is uncompressed, 0 if compressed.

Returns See *Error Codes*

int **wally_wif_to_public_key** (const char *wif, uint32_t prefix, unsigned char *bytes_out, size_t len, size_t *written)
Create a public key corresponding to a private key in Wallet Import Format.

Parameters

- **wif** – Private key in Wallet Import Format.
- **prefix** – Prefix byte to use, e.g. 0x80, 0xef.
- **bytes_out** – Destination for the resulting public key.
- **len** – The length of `bytes_out`.
- **written** – Destination for the number of bytes written to `bytes_out`.

Returns See [Variable Length Output Buffers](#)

int **wally_bip32_key_to_address** (const struct ext_key *hdkey, uint32_t flags, uint32_t version, char **output)

Create a legacy or wrapped SegWit address corresponding to a BIP32 key.

Parameters

- **hdkey** – The extended key to use.
- **flags** – WALLY_ADDRESS_TYPE_P2PKH for a legacy address, WALLY_ADDRESS_TYPE_P2SH_P2WPKH for P2SH-wrapped SegWit.
- **version** – Version byte to generate address, e.g. with Bitcoin: WALLY_ADDRESS_VERSION_P2PKH_MAINNET, WALLY_ADDRESS_VERSION_P2PKH_TESTNET, WALLY_ADDRESS_VERSION_P2SH_MAINNET and WALLY_ADDRESS_VERSION_P2SH_TESTNET.
- **output** – Destination for the resulting address string.

Returns See [Error Codes](#)

int **wally_bip32_key_to_addr_segwit** (const struct ext_key *hdkey, const char *addr_family, uint32_t flags, char **output)

Create a native SegWit address corresponding to a BIP32 key.

Parameters

- **hdkey** – The extended key to use.
- **addr_family** – Address family to generate, e.g. “bc” or “tb”.
- **flags** – For future use. Must be 0.
- **output** – Destination for the resulting segwit native address string.

Returns See [Error Codes](#)

int **wally_wif_to_address** (const char *wif, uint32_t prefix, uint32_t version, char **output)

Create a P2PKH address corresponding to a private key in Wallet Import Format.

Parameters

- **wif** – Private key in Wallet Import Format.
- **prefix** – Prefix byte to use, e.g. 0x80, 0xef.
- **version** – Version byte to generate address, e.g. WALLY_ADDRESS_VERSION_P2PKH_MAINNET, WALLY_ADDRESS_VERSION_P2PKH_TESTNET.
- **output** – Destination for the resulting address string.

Returns See [Error Codes](#)

int **wally_confidential_addr_to_addr** (const char *address, uint32_t prefix, char **output)

Extract the address from a confidential address.

Parameters

- **address** – The base58 encoded confidential address to extract the address from.
- **prefix** – The confidential address prefix byte, e.g. WALLY_CA_PREFIX_LIQUID.
- **output** – Destination for the resulting address string.

Returns See *Error Codes*

```
int wally_confidential_addr_to_ec_public_key (const char *address, uint32_t prefix, unsigned
                                             char *bytes_out, size_t len)
```

Extract the blinding public key from a confidential address.

Parameters

- **address** – The base58 encoded confidential address to extract the public key from.
- **prefix** – The confidential address prefix byte, e.g. WALLY_CA_PREFIX_LIQUID.
- **bytes_out** – Destination for the public key.
- **len** – Size of bytes_out. Must be EC_PUBLIC_KEY_LEN.

Returns See *Error Codes*

```
int wally_confidential_addr_from_addr (const char *address, uint32_t prefix, const unsigned
                                       char *pub_key, size_t pub_key_len, char **output)
```

Create a confidential address from an address and blinding public key.

Parameters

- **address** – The base58 encoded address to make confidential.
- **prefix** – The confidential address prefix byte, e.g. WALLY_CA_PREFIX_LIQUID.
- **pub_key** – The blinding public key to associate with address.
- **pub_key_len** – The length of pub_key in bytes. Must be EC_PUBLIC_KEY_LEN.
- **output** – Destination for the resulting address string.

Returns See *Error Codes*

```
int wally_confidential_addr_to_addr_segwit (const char *address, const char *confiden-
                                             tial_addr_family, const char *addr_family,
                                             char **output)
```

Extract the segwit native address from a confidential address.

Parameters

- **address** – The blech32 encoded confidential address to extract the address from.
- **confidential_addr_family** – The confidential address family of address.
- **addr_family** – The address family to generate.
- **output** – Destination for the resulting address string. The string returned should be freed using *wally_free_string*.

Returns See *Error Codes*

```
int wally_confidential_addr_segwit_to_ec_public_key (const char *address, const
                                                     char *confidential_addr_family, un-
                                                     signed char *bytes_out, size_t len)
```

Extract the blinding public key from a segwit confidential address.

Parameters

- **address** – The blech32 encoded confidential address to extract the public key from.
- **confidential_addr_family** – The confidential address family of address.
- **bytes_out** – Destination for the public key.
- **len** – Size of bytes_out. Must be EC_PUBLIC_KEY_LEN.

Returns See *Error Codes*

int **wally_confidential_addr_from_addr_segwit** (const char *address, const char *addr_family,
const char *confidential_addr_family, const
unsigned char *pub_key, size_t pub_key_len,
char **output)

Create a confidential address from a segwit native address and blinding public key.

Parameters

- **address** – The bech32 encoded address to make confidential.
- **addr_family** – The address family to generate.
- **confidential_addr_family** – The confidential address family to generate.
- **pub_key** – The blinding public key to associate with address.
- **pub_key_len** – The length of pub_key in bytes. Must be EC_PUBLIC_KEY_LEN.
- **output** – Destination for the resulting address string. The string returned should be freed using *wally_free_string*.

Returns See *Error Codes*

Bip32 Functions

int **bip32_key_free** (const struct ext_key *hdkey)

Free a key allocated by *bip32_key_from_seed_alloc*, *bip32_key_from_seed_custom* or *bip32_key_unserialize_alloc*.

Parameters

- **hdkey** – Key to free.

Returns See *Error Codes*

int **bip32_key_init** (uint32_t version, uint32_t depth, uint32_t child_num, const unsigned char *chain_code, size_t chain_code_len, const unsigned char *pub_key, size_t pub_key_len, const unsigned char *priv_key, size_t priv_key_len, const unsigned char *hash160, size_t hash160_len, const unsigned char *parent160, size_t parent160_len, struct ext_key *output)

Initialize a key.

Returns See *Error Codes*

int **bip32_key_init_alloc** (uint32_t version, uint32_t depth, uint32_t child_num, const unsigned char *chain_code, size_t chain_code_len, const unsigned char *pub_key, size_t pub_key_len, const unsigned char *priv_key, size_t priv_key_len, const unsigned char *hash160, size_t hash160_len, const unsigned char *parent160, size_t parent160_len, struct ext_key **output)

As per *bip32_key_init*, but allocates the key.

Returns See *Error Codes*

int **bip32_key_from_seed_custom** (const unsigned char *bytes, size_t bytes_len, uint32_t version, const unsigned char *hmac_key, size_t hmac_key_len, uint32_t flags, struct ext_key *output)

Create a new master extended key from entropy.

This creates a new master key, i.e. the root of a new HD tree. The entropy passed in may produce an invalid key. If this happens, *WALLY_ERROR* will be returned and the caller should retry with new entropy.

Parameters

- **bytes** – Entropy to use.
- **bytes_len** – Size of bytes in bytes. Must be one of BIP32_ENTROPY_LEN_128, BIP32_ENTROPY_LEN_256 or BIP32_ENTROPY_LEN_512.
- **version** – Either BIP32_VER_MAIN_PRIVATE or BIP32_VER_TEST_PRIVATE, indicating mainnet or testnet/regtest respectively.
- **hmac_key** – Custom data to HMAC-SHA512 with bytes before creating the key. Pass NULL to use the default BIP32 key of “Bitcoin seed”.
- **hmac_key_len** – Size of hmac_key in bytes, or 0 if hmac_key is NULL.
- **flags** – Either BIP32_FLAG_SKIP_HASH to skip hash160 calculation, or 0.
- **output** – Destination for the resulting master extended key.

Returns See *Error Codes*

int **bip32_key_from_seed** (const unsigned char *bytes, size_t bytes_len, uint32_t version, uint32_t flags, struct ext_key *output)

As per *bip32_key_from_seed_custom* With the default BIP32 seed.

Returns See *Error Codes*

int **bip32_key_from_seed_custom_alloc** (const unsigned char *bytes, size_t bytes_len, uint32_t version, const unsigned char *hmac_key, size_t hmac_key_len, uint32_t flags, struct ext_key **output)

As per *bip32_key_from_seed_custom*, but allocates the key. .. note:: The returned key should be freed with *bip32_key_free*.

Returns See *Error Codes*

int **bip32_key_from_seed_alloc** (const unsigned char *bytes, size_t bytes_len, uint32_t version, uint32_t flags, struct ext_key **output)

As per *bip32_key_from_seed*, but allocates the key. .. note:: The returned key should be freed with *bip32_key_free*.

Returns See *Error Codes*

int **bip32_key_serialize** (const struct ext_key *hdkey, uint32_t flags, unsigned char *bytes_out, size_t len)

Serialize an extended key to memory using BIP32 format.

Parameters

- **hdkey** – The extended key to serialize.
- **flags** – BIP32_FLAG_KEY_ Flags indicating which key to serialize. You can not serialize a private extended key from a public extended key.
- **bytes_out** – Destination for the serialized key.
- **len** – Size of bytes_out. Must be BIP32_SERIALIZED_LEN.

Returns See *Error Codes*

int **bip32_key_unserialize** (const unsigned char *bytes, size_t bytes_len, struct ext_key *output)

Un-serialize an extended key from memory.

Parameters

- **bytes** – Storage holding the serialized key.
- **bytes_len** – Size of bytes in bytes. Must be BIP32_SERIALIZED_LEN.

- **output** – Destination for the resulting extended key.

Returns See *Error Codes*

int **bip32_key_unserialize_alloc** (const unsigned char *bytes, size_t bytes_len, struct ext_key **output)

As per *bip32_key_unserialize*, but allocates the key.

Note: The returned key should be freed with *bip32_key_free*.

Returns See *Error Codes*

int **bip32_key_from_parent** (const struct ext_key *hdkey, uint32_t child_num, uint32_t flags, struct ext_key *output)

Create a new child extended key from a parent extended key.

Parameters

- **hdkey** – The parent extended key.
- **child_num** – The child number to create. Numbers greater than or equal to BIP32_INITIAL_HARDENED_CHILD represent hardened keys that cannot be created from public parent extended keys.
- **flags** – BIP32_FLAG_KEY_ Flags indicating the type of derivation wanted. You can not derive a private child extended key from a public parent extended key.
- **output** – Destination for the resulting child extended key.

Returns See *Error Codes*

int **bip32_key_from_parent_alloc** (const struct ext_key *hdkey, uint32_t child_num, uint32_t flags, struct ext_key **output)

As per *bip32_key_from_parent*, but allocates the key. .. note:: The returned key should be freed with *bip32_key_free*.

Returns See *Error Codes*

int **bip32_key_from_parent_path** (const struct ext_key *hdkey, const uint32_t *child_path, size_t child_path_len, uint32_t flags, struct ext_key *output)

Create a new child extended key from a parent extended key and a path.

Parameters

- **hdkey** – The parent extended key.
- **child_path** – The path of child numbers to create.
- **child_path_len** – The number of child numbers in *child_path*.
- **flags** – BIP32_FLAG_ Flags indicating the type of derivation wanted.
- **output** – Destination for the resulting child extended key.

Note: If *child_path* contains hardened child numbers, then *hdkey*

must be an extended private key or this function will fail.

Returns See *Error Codes*

```
int bip32_key_from_parent_path_alloc (const struct ext_key *hdkey, const uint32_t *child_path,
                                     size_t child_path_len, uint32_t flags, struct ext_key **output)
```

As per `bip32_key_from_parent_path`, but allocates the key. .. note:: The returned key should be freed with `bip32_key_free`.

Returns See *Error Codes*

```
int bip32_key_from_parent_path_str (const struct ext_key *hdkey, const char *path_str,
                                   uint32_t child_num, uint32_t flags, struct ext_key **output)
```

Create a new child extended key from a parent extended key and a path string.

Parameters

- **hdkey** – The parent extended key.
- **path_str** – The BIP32 path string of child numbers to create.
- **child_num** – The child number to use if `path_str` contains a * wildcard.
- **flags** – BIP32_FLAG_ Flags indicating the type of derivation wanted.
- **output** – Destination for the resulting child extended key.

Note: If `child_path` contains hardened child numbers, then `hdkey`

must be an extended private key or this function will fail.

Returns See *Error Codes*

```
int bip32_key_from_parent_path_str_n (const struct ext_key *hdkey, const char *path_str,
                                     size_t path_str_len, uint32_t child_num, uint32_t flags,
                                     struct ext_key *output)
```

Create a new child extended key from a parent extended key and a known-length path string.

See `bip32_key_from_parent_path_str`.

Returns See *Error Codes*

```
int bip32_key_from_parent_path_str_alloc (const struct ext_key *hdkey, const char *path_str,
                                          uint32_t child_num, uint32_t flags, struct
                                          ext_key **output)
```

As per `bip32_key_from_parent_path_str`, but allocates the key. .. note:: The returned key should be freed with `bip32_key_free`.

Returns See *Error Codes*

```
int bip32_key_from_parent_path_str_n_alloc (const struct ext_key *hdkey, const char *path_str,
                                             size_t path_str_len, uint32_t child_num,
                                             uint32_t flags, struct ext_key **output)
```

As per `bip32_key_from_parent_path_str_n`, but allocates the key. .. note:: The returned key should be freed with `bip32_key_free`.

Returns See *Error Codes*

```
int bip32_key_with_tweak_from_parent_path (const struct ext_key *hdkey, const
                                           uint32_t *child_path, size_t child_path_len,
                                           uint32_t flags, struct ext_key *output)
```

Derive the pub tweak from a parent extended key and a path.

Parameters

- **hdkey** – The parent extended key.
- **child_path** – The path of child numbers to create.
- **child_path_len** – The number of child numbers in `child_path`.
- **flags** – BIP32_FLAG_ Flags indicating the type of derivation wanted.
- **output** – Destination for the resulting key.

Returns See *Error Codes*

```
int bip32_key_with_tweak_from_parent_path_alloc (const struct ext_key *hdkey,
                                                const uint32_t *child_path,
                                                size_t child_path_len, uint32_t flags,
                                                struct ext_key **output)
```

As per `bip32_key_with_tweak_from_parent_path`, but allocates the key. .. note:: The returned key should be freed with `bip32_key_free`.

Returns See *Error Codes*

```
int bip32_key_to_base58 (const struct ext_key *hdkey, uint32_t flags, char **output)
```

Convert an extended key to base58.

Parameters

- **hdkey** – The extended key.
- **flags** – BIP32_FLAG_KEY_ Flags indicating which key to serialize. You can not serialize a private extended key from a public extended key.
- **output** – Destination for the resulting key in base58. The string returned should be freed using `wally_free_string`.

Returns See *Error Codes*

```
int bip32_key_from_base58 (const char *base58, struct ext_key *output)
```

Convert a base58 encoded extended key to an extended key.

Parameters

- **base58** – The extended key in base58.
- **output** – Destination for the resulting extended key.

Returns See *Error Codes*

```
int bip32_key_from_base58_n (const char *base58, size_t base58_len, struct ext_key *output)
```

Convert a known-length base58 encoded extended key to an extended key.

See `bip32_key_from_base58`.

Returns See *Error Codes*

```
int bip32_key_from_base58_alloc (const char *base58, struct ext_key **output)
```

As per `bip32_key_from_base58`, but allocates the key.

Note: The returned key should be freed with `bip32_key_free`.

Returns See *Error Codes*

```
int bip32_key_from_base58_n_alloc (const char *base58, size_t base58_len, struct ext_key **out-
                                  put)
```

As per `bip32_key_from_base58_n`, but allocates the key.

Note: The returned key should be freed with `bip32_key_free`.

Returns See *Error Codes*

int **bip32_key_strip_private_key** (struct ext_key *hdkey)

Converts a private extended key to a public extended key. Afterwards, only public child extended keys can be derived, and only the public serialization can be created. If the provided key is already public, nothing will be done.

Parameters

- **hdkey** – The extended key to convert.

Returns See *Error Codes*

int **bip32_key_get_fingerprint** (struct ext_key *hdkey, unsigned char *bytes_out, size_t len)

Get the BIP32 fingerprint for an extended key. Performs hash160 calculation if previously skipped with BIP32_FLAG_SKIP_HASH.

Parameters

- **hdkey** – The extended key.
- **bytes_out** – Destination for the fingerprint.
- **len** – Size of bytes_out. Must be BIP32_KEY_FINGERPRINT_LEN.

Returns See *Error Codes*

Bip38 Functions

int **bip38_raw_from_private_key**(const unsigned char *bytes, size_t bytes_len, const unsigned char *pass, size_t pass_len, uint32_t flags, unsigned char *bytes_out, size_t len)

Encode a private key in raw BIP 38 address format.

Parameters

- **bytes** – Private key to use.
- **bytes_len** – Size of **bytes** in bytes. Must be `EC_PRIVATE_KEY_LEN`.
- **pass** – Password for the encoded private key.
- **pass_len** – Length of **pass** in bytes.
- **flags** – **BIP38_KEY_** flags indicating desired behavior.
- **bytes_out** – Destination for the resulting raw BIP38 address.
- **len** – Size of **bytes_out**. Must be `BIP38_SERIALIZED_LEN`.

Returns See [Error Codes](#)

int **bip38_from_private_key**(const unsigned char *bytes, size_t bytes_len, const unsigned char *pass, size_t pass_len, uint32_t flags, char **output)

Encode a private key in BIP 38 address format.

Parameters

- **bytes** – Private key to use.
- **bytes_len** – Size of **bytes** in bytes. Must be `EC_PRIVATE_KEY_LEN`.
- **pass** – Password for the encoded private key.
- **pass_len** – Length of **pass** in bytes.
- **flags** – **BIP38_KEY_** flags indicating desired behavior.
- **output** – Destination for the resulting BIP38 address.

Returns See *Error Codes*

int **bip38_raw_to_private_key** (const unsigned char *bytes, size_t bytes_len, const unsigned char *pass, size_t pass_len, uint32_t flags, unsigned char *bytes_out, size_t len)
Decode a raw BIP 38 address to a private key.

Parameters

- **bytes** – Raw BIP 38 address to decode.
- **bytes_len** – Size of bytes in bytes. Must be BIP38_SERIALIZED_LEN.
- **pass** – Password for the encoded private key.
- **pass_len** – Length of pass in bytes.
- **flags** – **BIP38_KEY_** flags indicating desired behavior.
- **bytes_out** – Destination for the resulting private key.
- **len** – Size of bytes_out. Must be EC_PRIVATE_KEY_LEN.

Returns See *Error Codes*

int **bip38_to_private_key** (const char *bip38, const unsigned char *pass, size_t pass_len, uint32_t flags, unsigned char *bytes_out, size_t len)
Decode a BIP 38 address to a private key.

Parameters

- **bip38** – BIP 38 address to decode.
- **pass** – Password for the encoded private key.
- **pass_len** – Length of pass in bytes.
- **flags** – **BIP38_KEY_** flags indicating desired behavior.
- **bytes_out** – Destination for the resulting private key.
- **len** – Size of bytes_out. Must be EC_PRIVATE_KEY_LEN.

Returns See *Error Codes*

int **bip38_raw_get_flags** (const unsigned char *bytes, size_t bytes_len, size_t *written)
Get compression and/or EC mult flags.

Parameters

- **bytes** – Raw BIP 38 address to get the flags from.
- **bytes_len** – Size of bytes in bytes. Must be BIP38_SERIALIZED_LEN.
- **written** – **BIP38_KEY_** flags indicating behavior.

Returns See *Error Codes*

int **bip38_get_flags** (const char *bip38, size_t *written)
Get compression and/or EC mult flags.

Parameters

- **bip38** – BIP 38 address to get the flags from.
- **written** – **BIP38_KEY_** flags indicating behavior.

Returns See *Error Codes*

Bip39 Functions

int **bip39_get_languages** (char ***output*)

Get the list of default supported languages.

..note:: The string returned should be freed using *wally_free_string*.

Returns See *Error Codes*

int **bip39_get_wordlist** (const char **lang*, struct words ***output*)

Get the default word list for a language.

Parameters

- **lang** – Language to use. Pass NULL to use the default English list.
- **output** – Destination for the resulting word list.

Note: The returned structure should not be freed or modified.

Returns See *Error Codes*

int **bip39_get_word** (const struct words **w*, size_t *index*, char ***output*)

Get the ‘index’th word from a word list.

Parameters

- **w** – Word list to use. Pass NULL to use the default English list.
- **index** – The 0-based index of the word in w.
- **output** – Destination for the resulting word.

The string returned should be freed using *wally_free_string*.

Returns See *Error Codes*

int **bip39_mnemonic_from_bytes** (const struct words *w, const unsigned char *bytes, size_t bytes_len, char **output)

Generate a mnemonic sentence from the entropy in bytes.

Parameters

- **w** – Word list to use. Pass NULL to use the default English list.
- **bytes** – Entropy to convert.
- **bytes_len** – The length of bytes in bytes.
- **output** – Destination for the resulting mnemonic sentence.

Note: The string returned should be freed using *wally_free_string*.

Returns See *Error Codes*

int **bip39_mnemonic_to_bytes** (const struct words *w, const char *mnemonic, unsigned char *bytes_out, size_t len, size_t *written)

Convert a mnemonic sentence into entropy at bytes_out.

Parameters

- **w** – Word list to use. Pass NULL to use the default English list.
- **mnemonic** – Mnemonic to convert.
- **bytes_out** – Where to store the resulting entropy.
- **n** – Size of bytes_out. Passing BIP39_ENTROPY_MAX_LEN will ensure the buffer is large enough.
- **written** – Destination for the number of bytes written to bytes_out.

Returns See *Variable Length Output Buffers*

int **bip39_mnemonic_validate** (const struct words *w, const char *mnemonic)

Validate the checksum embedded in a mnemonic sentence.

Parameters

- **w** – Word list to use. Pass NULL to use the default English list.
- **mnemonic** – Mnemonic to validate.

Returns See *Error Codes*

int **bip39_mnemonic_to_seed** (const char *mnemonic, const char *passphrase, unsigned char *bytes_out, size_t len, size_t *written)

Convert a mnemonic into a binary seed.

Parameters

- **mnemonic** – Mnemonic to convert.
- **passphrase** – Mnemonic passphrase or NULL if no passphrase is needed.
- **bytes_out** – The destination for the binary seed.
- **len** – Size of bytes_out. Must be BIP39_SEED_LEN_512.
- **written** – Destination for the number of bytes written to bytes_out.

Returns See *Variable Length Output Buffers*

```
int bip39_mnemonic_to_seed512(const char *mnemonic, const char *passphrase, unsigned  
                               char *bytes_out, size_t len)
```

Convert a mnemonic into a binary seed of 512 bits.

Parameters

- **mnemonic** – Mnemonic to convert.
- **passphrase** – Mnemonic passphrase or NULL if no passphrase is needed.
- **bytes_out** – The destination for the binary seed.
- **len** – Size of bytes_out. Must be BIP39_SEED_LEN_512.

..note:: Identical to *bip39_mnemonic_to_seed* but returns a fixed size buffer.

Returns See *Error Codes*

Map Functions

int **wally_map_init** (size_t *allocation_len*, wally_map_verify_fn_t *verify_fn*, struct wally_map **output*)
Initialize a new map.

Parameters

- **allocation_len** – The number of items to allocate space for.
- **verify_fn** – A function to verify items before storing, or NULL.
- **output** – Map to initialize.

Returns See [Error Codes](#)

int **wally_map_init_alloc** (size_t *allocation_len*, wally_map_verify_fn_t *verify_fn*, struct wally_map ***output*)
Allocate and initialize a new map.

Parameters

- **allocation_len** – The number of items to allocate space for.
- **verify_fn** – A function to verify items before storing, or NULL.
- **output** – Destination for the new map.

Returns See [Error Codes](#)

int **wally_map_free** (struct wally_map **map_in*)
Free a map allocated by [wally_map_init_alloc](#).

Parameters

- **map_in** – The map to free.

Returns See [Error Codes](#)

int **wally_map_clear** (struct wally_map **map_in*)
Remove all entries from a map.

Parameters

- **map_in** – The map to clear.

Note: This function frees all pre-allocated memory, and thus can be used to free a map initialised with `wally_map_init` without freeing the map struct itself.

Returns See *Error Codes*

int **wally_map_add** (struct wally_map *map_in, const unsigned char *key, size_t key_len, const unsigned char *value, size_t value_len)
Add an item to a map.

Parameters

- **map_in** – The map to add to.
- **key** – The key to add.
- **key_len** – Length of key in bytes.
- **value** – The value to add.
- **value_len** – Length of value in bytes.

Note: If the key given is already in the map, this call succeeds without altering the map.

Returns See *Error Codes*

int **wally_map_add_integer** (struct wally_map *map_in, uint32_t key, const unsigned char *value, size_t value_len)
Add an item to a map keyed by an integer.
As per `wally_map_add`, using an integer key.

Returns See *Error Codes*

int **wally_map_replace** (struct wally_map *map_in, const unsigned char *key, size_t key_len, const unsigned char *value, size_t value_len)
Add an item to a map, replacing it if already present.
See `wally_map_add`.

Returns See *Error Codes*

int **wally_map_replace_integer** (struct wally_map *map_in, uint32_t key, const unsigned char *value, size_t value_len)
Add an item to a map keyed by an integer, replacing it if already present.
See `wally_map_add_integer`.

Returns See *Error Codes*

int **wally_map_remove** (struct wally_map *map_in, const unsigned char *key, size_t key_len)
Remove an item from a map.

Parameters

- **map_in** – The map to remove from.
- **key** – The key to add.
- **key_len** – Length of key in bytes.

Returns See *Error Codes*

int **wally_map_remove_integer** (struct wally_map *map_in, uint32_t key)
Remove an item from a map keyed by an integer.

See *wally_map_remove_integer*.

Returns See *Error Codes*

int **wally_map_find_from** (const struct wally_map *map_in, size_t index, const unsigned char *key,
size_t key_len, size_t *written)
Find an item in a map from a given position onwards.

Parameters

- **map_in** – The map to find key in.
- **index** – The zero-based index of the item to start searching from.
- **key** – The key to find.
- **key_len** – Length of key in bytes.
- **written** – On success, set to zero if the item is not found, otherwise the index of the item plus one.

Returns See *Error Codes*

int **wally_map_find** (const struct wally_map *map_in, const unsigned char *key, size_t key_len,
size_t *written)
Find an item in a map.

Parameters

- **map_in** – The map to find key in.
- **key** – The key to find.
- **key_len** – Length of key in bytes.
- **written** – On success, set to zero if the item is not found, otherwise the index of the item plus one.

Returns See *Error Codes*

int **wally_map_find_integer** (const struct wally_map *map_in, uint32_t key, size_t *written)
Find an item in a map keyed by an integer.

As per *wally_map_find*, using an integer key.

Returns See *Error Codes*

const struct wally_map_item ***wally_map_get** (const struct wally_map *map_in, const unsigned char *key,
size_t key_len)
Find an item in a map.

Parameters

- **map_in** – The map to find key in.
- **key** – The key to find.
- **key_len** – Length of key in bytes.

Note: This is a non-standard call for low-level use. It returns the map item directly without copying, or NULL if not found/an error occurs.

const struct wally_map_item ***wally_map_get_integer** (const struct wally_map *map_in, uint32_t key)
Find an item in a map keyed by an integer.

As per *wally_map_get*, using an integer key.

int **wally_map_get_num_items** (const struct wally_map *map_in, size_t *written)
Get the number of key/value items in a map.

Parameters

- **map_in** – The map to return the number of items from.
- **written** – Destination for the number of items.

Returns See *Error Codes*

int **wally_map_get_item_key_length** (const struct wally_map *map_in, size_t index, size_t *written)
Get the length of an items key in a map.

Parameters

- **map_in** – The map to return the items key length from.
- **index** – The zero-based index of the item whose key length to return.
- **written** – Destination for the length of the items key in bytes.

Note: Returns 0 if the items key is an integer.

Returns See *Error Codes*

int **wally_map_get_item_key** (const struct wally_map *map_in, size_t index, unsigned char *bytes_out,
size_t len, size_t *written)
Return an items key from a map.

Parameters

- **map_in** – The map to return the items key from.
- **index** – The zero-based index of the item whose key to return.
- **bytes_out** – Destination for the resulting data.
- **len** – The length of bytes_out in bytes.
- **written** – Destination for the number of bytes written to bytes_out.

Note: Returns WALLY_ERROR if the items key is an integer.

Returns See *Variable Length Output Buffers*

int **wally_map_get_item_integer_key** (const struct wally_map *map_in, size_t index, size_t *written)
Return an items integer key from a map.

Parameters

- **map_in** – The map to return the items key from.
- **index** – The zero-based index of the item whose key to return.
- **written** – Destination for the items integer key.

Note: Returns `WALLY_ERROR` if the items key is not an integer.

Returns See [Error Codes](#)

int **wally_map_get_item_length** (const struct wally_map *map_in, size_t index, size_t *written)
Get the length of an item in a map.

Parameters

- **map_in** – The map to return the item length from.
- **index** – The zero-based index of the item whose length to return.
- **written** – Destination for the length of the item in bytes.

Returns See [Error Codes](#)

int **wally_map_get_item** (const struct wally_map *map_in, size_t index, unsigned char *bytes_out,
size_t len, size_t *written)
Return an item from a map.

Parameters

- **map_in** – The map to return the item from.
- **index** – The zero-based index of the item to return.
- **bytes_out** – Destination for the resulting data.
- **len** – The length of bytes_out in bytes.
- **written** – Destination for the number of bytes written to bytes_out.

Returns See [Variable Length Output Buffers](#)

int **wally_map_sort** (struct wally_map *map_in, uint32_t flags)
Sort the items in a map.

Parameters

- **map_in** – The map to sort.
- **flags** – Flags controlling sorting. Must be 0.

Returns See [Error Codes](#)

int **wally_map_combine** (struct wally_map *map_in, const struct wally_map *source)
Combine the items from a source map into another map.

Parameters

- **map_in** – the destination to combine into.
- **source** – the source to copy items from.

Note: If this call fails, map_in may be left partially updated.

Returns See [Error Codes](#)

int **wally_map_assign** (struct wally_map *map_in, const struct wally_map *source)
Replace a maps contents with another map.

Parameters

- **map_in** – the destination to combine into.
- **source** – the source to copy items from.

Note: If this call fails, `map_in` is left untouched.

Returns See *Error Codes*

int **wally_map_find_bip32_public_key_from** (const struct wally_map *map_in, size_t index, const struct ext_key *hdkey, size_t *written)

Find an item in a public-key keyed map given a BIP32 derived key.

Parameters

- **map_in** – The map to find the public key of hdkey in.
- **index** – The zero-based index of the item to start searching from.
- **hdkey** – The BIP32 key to find.
- **written** – On success, set to zero if the item is not found, otherwise the index of the item plus one.

Note: This function searches for the compressed, x-only and then uncompressed public keys, in order. The caller can determine which by checking the length of the map item when an item is found.

Returns See *Error Codes*

int **wally_map_keypath_get_bip32_key_from_alloc** (const struct wally_map *map_in, size_t index, const struct ext_key *hdkey, struct ext_key **output)

Return a BIP32 derived key matching the keypath of a parent in a map.

Parameters

- **map_in** – The map to search for derived keys of hdkey in.
- **index** – The zero-based index of the item to start searching from.
- **hdkey** – The BIP32 parent key to derive matches from.
- **output** – Destination for the resulting derived key, if any.

Note: This function searches for keys in the map that are children of hdkey. If one is found, the resulting privately derived key is returned. If no key is found, *output is set to NULL and WALLY_OK is returned.

Returns See *Error Codes*

int **wally_keypath_bip32_verify** (const unsigned char *key, size_t key_len, const unsigned char *val, size_t val_len)

Verify a PSBT keypath keyed by a serialized bip32 extended public key.

Parameters

- **key** – An extended public key in bip32 format.

- **key_len** – Length of *key* in bytes. Must be BIP32_SERIALIZED_LEN.
- **val** – The 4 byte PSBT serialized master key fingerprint followed by the serialized path.
- **val_len** – Length of *val* in bytes.

Returns See *Error Codes*

int **wally_keypath_public_key_verify** (const unsigned char **key*, size_t *key_len*, const unsigned char **val*, size_t *val_len*)

Verify a PSBT keypath keyed by a compressed or uncompressed public key.

Parameters

- **key** – Public key bytes.
- **key_len** – Length of *key* in bytes. Must be EC_PUBLIC_KEY_UNCOMPRESSED_LEN or BIP32_SERIALIZED_LEN.
- **val** – The 4 byte PSBT serialized master key fingerprint followed by the serialized path.
- **val_len** – Length of *val* in bytes.

Returns See *Error Codes*

int **wally_keypath_xonly_public_key_verify** (const unsigned char **key*, size_t *key_len*, const unsigned char **val*, size_t *val_len*)

Verify a PSBT keypath keyed by an x-only public key.

Parameters

- **key** – Public key bytes.
- **key_len** – Length of *key* in bytes. Must be EC_XONLY_PUBLIC_KEY_LEN.
- **val** – The 4 byte PSBT serialized master key fingerprint followed by the serialized path.
- **val_len** – Length of *val* in bytes.

Returns See *Error Codes*

int **wally_map_keypath_bip32_init_alloc** (size_t *allocation_len*, struct wally_map ***output*)

Allocate and initialize a new BIP32 keypath map.

Parameters

- **allocation_len** – The number of items to allocate space for.
- **output** – Destination for the new map.

Returns See *Error Codes*

int **wally_map_keypath_public_key_init_alloc** (size_t *allocation_len*, struct wally_map ***output*)

Allocate and initialize a new public key keypath map.

Parameters

- **allocation_len** – The number of items to allocate space for.
- **output** – Destination for the new map.

Returns See *Error Codes*

int **wally_map_keypath_add** (struct wally_map **map_in*, const unsigned char **pub_key*, size_t *pub_key_len*, const unsigned char **fingerprint*, size_t *fingerprint_len*, const uint32_t **child_path*, size_t *child_path_len*)

Convert and add a public key and path to a keypath map.

Parameters

- **map_in** – The keypath map to add to.
- **pub_key** – The public key or extended public key to add.
- **pub_key_len** – Length of `pub_key` in bytes. Must be `BIP32_SERIALIZED_LEN` for an extended bip32 public key, or `EC_PUBLIC_KEY_UNCOMPRESSED_LEN` or `EC_PUBLIC_KEY_LEN` for a public key.
- **fingerprint** – The master key fingerprint for the pubkey.
- **fingerprint_len** – Length of fingerprint in bytes. Must be `BIP32_KEY_FINGERPRINT_LEN`.
- **child_path** – The BIP32 derivation path for the pubkey.
- **child_path_len** – The number of items in `child_path`.

Returns See *Error Codes*

int **wally_keypath_get_fingerprint** (const unsigned char *val, size_t val_len, unsigned char *bytes_out, size_t len)

Get the key fingerprint from a PSBT keypath's serialized value.

Parameters

- **val** – The serialized keypath value as stored in a keypath map.
- **val_len** – Length of `val` in bytes.
- **bytes_out** – Destination for the fingerprint.
- **len** – Size of `bytes_out`. Must be `BIP32_KEY_FINGERPRINT_LEN`.

Returns See *Error Codes*

int **wally_map_keypath_get_item_fingerprint** (const struct wally_map *map_in, size_t index, unsigned char *bytes_out, size_t len)

Return an item's key fingerprint from a keypath map.

Parameters

- **map_in** – The map to return the item's fingerprint from.
- **index** – The zero-based index of the item whose key fingerprint to return.
- **bytes_out** – Destination for the resulting data.
- **len** – Size of `bytes_out`. Must be `BIP32_KEY_FINGERPRINT_LEN`.

Note: The same key fingerprint may be present in a keypath map more than once.

Returns See *Error Codes*

int **wally_keypath_get_path_len** (const unsigned char *val, size_t val_len, size_t *written)

Get the path length from a PSBT keypath's serialized value.

Parameters

- **val** – The serialized keypath value as stored in a keypath map.
- **val_len** – Length of `val` in bytes.
- **written** – Destination for the items path length.

Returns See [Error Codes](#)

int **wally_map_keypath_get_item_path_len**(const struct wally_map *map_in, size_t index,
size_t *written)
Get the length of an item's key path from a keypath map.

Parameters

- **map_in** – The map to return the item's path length from.
- **index** – The zero-based index of the item whose path length to return.
- **written** – Destination for the items path length.

Returns See [Error Codes](#)

int **wally_keypath_get_path**(const unsigned char *val, size_t val_len, uint32_t *child_path_out,
size_t child_path_out_len, size_t *written)
Get the path from a PSBT keypath's serialized value.

Parameters

- **val** – The serialized keypath value as stored in a keypath map.
- **val_len** – Length of val in bytes.
- **child_path_out** – Destination for the resulting path.
- **child_path_out_len** – The number of items in child_path_out.
- **written** – Destination for the number of items written to child_path_out.

Note: If The path is longer than child_path_out_len, WALLY_OK is returned and written contains the length required. It is valid for a path to be zero-length.

Note: This function should be used to read paths from serialized keypath values to prevent endianness issues on big-endian hosts.

Returns See [Error Codes](#)

int **wally_map_keypath_get_item_path**(const struct wally_map *map_in, size_t index,
uint32_t *child_path_out, size_t child_path_out_len,
size_t *written)
Get the path from a PSBT keypath's serialized value.

Parameters

- **map_in** – The map to return the item's path from.
- **index** – The zero-based index of the item whose path to return.
- **child_path_out** – Destination for the resulting path.
- **child_path_out_len** – The number of items in child_path_out.
- **written** – Destination for the number of items written to child_path_out.

Note: See the notes for [wally_keypath_get_path](#).

Returns See [Error Codes](#)

int **wally_map_hash_preimage_verify** (const unsigned char *key, size_t key_len, const unsigned char *val, size_t val_len)

Verify a preimage map key and value pair.

Parameters

- **key** – The preimage hash, prefixed by a hash type byte.
- **key_len** – Length of key in bytes.
- **val** – The preimage data hashed to produce key.
- **val_len** – Length of val in bytes.

Note: Multiple preimage types are stored in the same map, prefixed by a leading byte. The exact format of storage is implementation dependent and may change in the future.

Returns See *Error Codes*

int **wally_map_preimage_init_alloc** (size_t allocation_len, struct wally_map **output)

Allocate and initialize a new preimage map.

Parameters

- **allocation_len** – The number of items to allocate space for.
- **output** – Destination for the new map.

Returns See *Error Codes*

int **wally_map_preimage_ripemd160_add** (struct wally_map *map_in, const unsigned char *value, size_t value_len)

Add a ripemd160 preimage to a preimage map.

Parameters

- **map_in** – The preimage map to add to.
- **value** – The data to store.
- **value_len** – Length of value in bytes.

Returns See *Error Codes*

int **wally_map_preimage_sha256_add** (struct wally_map *map_in, const unsigned char *value, size_t value_len)

Add a sha256 preimage to a preimage map.

Parameters

- **map_in** – The preimage map to add to.
- **value** – The data to store.
- **value_len** – Length of value in bytes.

Returns See *Error Codes*

int **wally_map_preimage_hash160_add** (struct wally_map *map_in, const unsigned char *value, size_t value_len)

Add a hash160 preimage to a preimage map.

Parameters

- **map_in** – The preimage map to add to.

- **value** – The data to store.
- **value_len** – Length of `value` in bytes.

Returns See *Error Codes*

int **wally_map_preimage_sha256d_add**(struct wally_map *map_in, const unsigned char *value,
size_t value_len)

Add a sha256d preimage to a preimage map.

Parameters

- **map_in** – The preimage map to add to.
- **value** – The data to store.
- **value_len** – Length of `value` in bytes.

Returns See *Error Codes*

Psbt Functions

int **wally_psbt_input_set_previous_txid** (struct wally_psbt_input *input, const unsigned char *tx-hash, size_t txhash_len)

Set the previous txid in an input.

Parameters

- **input** – The input to update.
- **txhash** – The previous hash for this input.
- **txhash_len** – Length of txhash in bytes. Must be WALLY_TXHASH_LEN.

Returns See [Error Codes](#)

int **wally_psbt_input_set_output_index** (struct wally_psbt_input *input, uint32_t index)

Set the output index in an input.

Parameters

- **input** – The input to update.
- **index** – The index of the spent output for this input.

Returns See [Error Codes](#)

int **wally_psbt_input_set_sequence** (struct wally_psbt_input *input, uint32_t sequence)

Set the sequence number in an input.

Parameters

- **input** – The input to update.
- **sequence** – The sequence number for this input.

Returns See [Error Codes](#)

int **wally_psbt_input_clear_sequence** (struct wally_psbt_input *input)

Clear the sequence number in an input.

Parameters

- **input** – The input to update.

Returns See *Error Codes*

int **wally_psbt_input_set_utxo** (struct wally_psbt_input *input, const struct wally_tx *utxo)
Set the utxo in an input.

Parameters

- **input** – The input to update.
- **utxo** – The (non witness) utxo for this input if it exists.

Returns See *Error Codes*

int **wally_psbt_input_set_witness_utxo** (struct wally_psbt_input *input, const struct wally_tx_output *witness_utxo)
Set the witness_utxo in an input.

Parameters

- **input** – The input to update.
- **witness_utxo** – The witness utxo for this input if it exists.

Returns See *Error Codes*

int **wally_psbt_input_set_witness_utxo_from_tx** (struct wally_psbt_input *input, const struct wally_tx *utxo, uint32_t index)
Set the witness_utxo in an input from a transaction output.

Parameters

- **input** – The input to update.
- **utxo** – The transaction containing the output to add.
- **index** – The output index in utxo to add.

Returns See *Error Codes*

int **wally_psbt_input_set_redeem_script** (struct wally_psbt_input *input, const unsigned char *script, size_t script_len)
Set the redeem_script in an input.

Parameters

- **input** – The input to update.
- **script** – The redeem script for this input.
- **script_len** – Length of script in bytes.

Returns See *Error Codes*

int **wally_psbt_input_set_witness_script** (struct wally_psbt_input *input, const unsigned char *script, size_t script_len)
Set the witness_script in an input.

Parameters

- **input** – The input to update.
- **script** – The witness script for this input.
- **script_len** – Length of script in bytes.

Returns See *Error Codes*

int **wally_psbtx_input_set_final_scriptsig** (struct wally_psbtx_input *input, const unsigned char *final_scriptsig, size_t final_scriptsig_len)

Set the final_scriptsig in an input.

Parameters

- **input** – The input to update.
- **final_scriptsig** – The scriptSig for this input.
- **final_scriptsig_len** – Length of final_scriptsig in bytes.

Returns See *Error Codes*

int **wally_psbtx_input_set_final_witness** (struct wally_psbtx_input *input, const struct wally_tx_witness_stack *witness)

Set the final witness in an input.

Parameters

- **input** – The input to update.
- **witness** – The witness stack for the input, or NULL if not present.

Returns See *Error Codes*

int **wally_psbtx_input_set_keypaths** (struct wally_psbtx_input *input, const struct wally_map *map_in)

Set the keypaths in an input.

Parameters

- **input** – The input to update.
- **map_in** – The HD keypaths for this input.

Returns See *Error Codes*

int **wally_psbtx_input_find_keypath** (struct wally_psbtx_input *input, const unsigned char *pub_key, size_t pub_key_len, size_t *written)

Find a keypath matching a pubkey in an input.

Parameters

- **input** – The input to search in.
- **pub_key** – The pubkey to find.
- **pub_key_len** – Length of pub_key in bytes. Must be EC_PUBLIC_KEY_UNCOMPRESSED_LEN or EC_PUBLIC_KEY_LEN.
- **written** – On success, set to zero if the item is not found, otherwise the index of the item plus one.

Returns See *Error Codes*

int **wally_psbtx_input_keypath_add** (struct wally_psbtx_input *input, const unsigned char *pub_key, size_t pub_key_len, const unsigned char *fingerprint, size_t fingerprint_len, const uint32_t *child_path, size_t child_path_len)

Convert and add a pubkey/keypath to an input.

Parameters

- **input** – The input to add to.
- **pub_key** – The pubkey to add.

- **pub_key_len** – Length of pub_key in bytes. Must be EC_PUBLIC_KEY_UNCOMPRESSED_LEN or EC_PUBLIC_KEY_LEN.
- **fingerprint** – The master key fingerprint for the pubkey.
- **fingerprint_len** – Length of fingerprint in bytes. Must be BIP32_KEY_FINGERPRINT_LEN.
- **child_path** – The BIP32 derivation path for the pubkey.
- **child_path_len** – The number of items in child_path.

Returns See *Error Codes*

int **wally_psbtx_input_set_signatures** (struct wally_psbtx_input *input, const struct wally_map *map_in)

Set the partial signatures in an input.

Parameters

- **input** – The input to update.
- **map_in** – The partial signatures for this input.

Returns See *Error Codes*

int **wally_psbtx_input_find_signature** (struct wally_psbtx_input *input, const unsigned char *pub_key, size_t pub_key_len, size_t *written)

Find a partial signature matching a pubkey in an input.

Parameters

- **input** – The input to search in.
- **pub_key** – The pubkey to find.
- **pub_key_len** – Length of pub_key in bytes. Must be EC_PUBLIC_KEY_UNCOMPRESSED_LEN or EC_PUBLIC_KEY_LEN.
- **written** – On success, set to zero if the item is not found, otherwise the index of the item plus one.

Returns See *Error Codes*

int **wally_psbtx_input_add_signature** (struct wally_psbtx_input *input, const unsigned char *pub_key, size_t pub_key_len, const unsigned char *sig, size_t sig_len)

Add a pubkey/partial signature item to an input.

Parameters

- **input** – The input to add the partial signature to.
- **pub_key** – The pubkey to find.
- **pub_key_len** – Length of pub_key in bytes. Must be EC_PUBLIC_KEY_UNCOMPRESSED_LEN or EC_PUBLIC_KEY_LEN.
- **sig** – The DER-encoded signature plus sighash byte to add.
- **sig_len** – The length of sig in bytes.

Returns See *Error Codes*

int **wally_psbtx_input_set_unknowns** (struct wally_psbtx_input *input, const struct wally_map *map_in)

Set the unknown values in an input.

Parameters

- **input** – The input to update.
- **map_in** – The unknown key value pairs for this input.

Returns See *Error Codes*

int **wally_psbtx_input_find_unknown** (struct wally_psbtx_input *input, const unsigned char *key, size_t key_len, size_t *written)

Find an unknown item matching a key in an input.

Parameters

- **input** – The input to search in.
- **key** – The key to find.
- **key_len** – Length of key in bytes.
- **written** – On success, set to zero if the item is not found, otherwise the index of the item plus one.

Returns See *Error Codes*

int **wally_psbtx_input_set_sighash** (struct wally_psbtx_input *input, uint32_t sighash)

Set the sighash type in an input.

Parameters

- **input** – The input to update.
- **sighash** – The sighash type for this input.

Returns See *Error Codes*

int **wally_psbtx_input_set_required_locktime** (struct wally_psbtx_input *input, uint32_t required_locktime)

Set the required lock time in an input.

Parameters

- **input** – The input to update.
- **required_locktime** – The required locktime for this input.

Returns See *Error Codes*

int **wally_psbtx_input_clear_required_locktime** (struct wally_psbtx_input *input)

Clear the required lock time in an input.

Parameters

- **input** – The input to update.

Returns See *Error Codes*

int **wally_psbtx_input_set_required_lockheight** (struct wally_psbtx_input *input, uint32_t required_lockheight)

Set the required lock height in an input.

Parameters

- **input** – The input to update.
- **required_lockheight** – The required locktime for this input.

Returns See *Error Codes*

int **wally_psbt_input_clear_required_lockheight** (struct wally_psbt_input *input)
Clear the required lock height in an input.

Parameters

- **input** – The input to update.

Returns See *Error Codes*

int **wally_psbt_input_set_amount** (struct wally_psbt_input *input, uint64_t amount)
Set the unblinded amount in an input.

Parameters

- **input** – The input to update.
- **amount** – The amount of the input.

Note: This operates on the PSET field PSBT_ELEMENTS_IN_EXPLICIT_VALUE.

Returns See *Error Codes*

int **wally_psbt_input_get_amount_rangeproof** (const struct wally_psbt_input *input, unsigned char *bytes_out, size_t len, size_t *written)
Get the explicit amount rangeproof from an input.

Parameters

- **input** – The input to get from.
- **bytes_out** – Destination for the explicit amount rangeproof.
- **len** – Size of bytes_out in bytes.
- **written** – Destination for the number of bytes written to bytes_out. Will be zero if the value is not present.

Note: This operates on the PSET field PSBT_ELEMENTS_IN_VALUE_PROOF.

Returns See *Variable Length Output Buffers*

int **wally_psbt_input_get_amount_rangeproof_len** (const struct wally_psbt_input *input, size_t *written)
Get the length of the explicit amount rangeproof from an input.

Parameters

- **input** – The input to get from.
- **written** – Destination for the length, or zero if not present.

Returns See *Error Codes*

int **wally_psbt_input_set_amount_rangeproof** (struct wally_psbt_input *input, const unsigned char *rangeproof, size_t rangeproof_len)
Set the explicit amount rangeproof in an input.

Parameters

- **input** – The input to update.

- **rangeproof** – The explicit amount rangeproof.
- **rangeproof_len** – Size of rangeproof in bytes.

Returns See *Error Codes*

int **wally_psbt_input_clear_amount_rangeproof** (struct wally_psbt_input *input)

Clear the explicit amount rangeproof in an input.

Parameters

- **input** – The input to update.

Returns See *Error Codes*

int **wally_psbt_input_get_asset** (const struct wally_psbt_input *input, unsigned char *bytes_out,
size_t len, size_t *written)

Get the explicit asset tag from an input.

Parameters

- **input** – The input to get from.
- **bytes_out** – Destination for the explicit asset tag.
- **len** – Size of bytes_out in bytes.
- **written** – Destination for the number of bytes written to bytes_out. Will be zero if the value is not present.

Note: This operates on the PSET field PSBT_ELEMENTS_IN_EXPLICIT_ASSET.

Returns See *Variable Length Output Buffers*

int **wally_psbt_input_get_asset_len** (const struct wally_psbt_input *input, size_t *written)

Get the length of the explicit asset tag from an input.

Parameters

- **input** – The input to get from.
- **written** – Destination for the length, or zero if not present.

Returns See *Error Codes*

int **wally_psbt_input_set_asset** (struct wally_psbt_input *input, const unsigned char *asset,
size_t asset_len)

Set the explicit asset tag in an input.

Parameters

- **input** – The input to update.
- **asset** – The explicit asset tag.
- **asset_len** – Size of asset in bytes. Must be ASSET_TAG_LEN.

Returns See *Error Codes*

int **wally_psbt_input_clear_asset** (struct wally_psbt_input *input)

Clear the explicit asset tag in an input.

Parameters

- **input** – The input to update.

Returns See *Error Codes*

```
int wally_psbt_input_get_asset_surjectionproof (const struct wally_psbt_input *input,
                                                unsigned char *bytes_out, size_t len,
                                                size_t *written)
```

Get the explicit asset surjection proof from an input.

Parameters

- **input** – The input to get from.
- **bytes_out** – Destination for the explicit asset surjection proof.
- **len** – Size of `bytes_out` in bytes.
- **written** – Destination for the number of bytes written to `bytes_out`. Will be zero if the value is not present.

Note: This operates on the PSET field `PSBT_ELEMENTS_IN_ASSET_PROOF`.

Returns See *Variable Length Output Buffers*

```
int wally_psbt_input_get_asset_surjectionproof_len (const struct wally_psbt_input *input,
                                                    size_t *written)
```

Get the length of the explicit asset surjection proof from an input.

Parameters

- **input** – The input to get from.
- **written** – Destination for the length, or zero if not present.

Returns See *Error Codes*

```
int wally_psbt_input_set_asset_surjectionproof (struct wally_psbt_input *input, const unsigned char *surjectionproof, size_t surjectionproof_len)
```

Set the explicit asset surjection proof in an input.

Parameters

- **input** – The input to update.
- **surjectionproof** – The explicit asset surjection proof.
- **surjectionproof_len** – Size of `surjectionproof` in bytes.

Returns See *Error Codes*

```
int wally_psbt_input_clear_asset_surjectionproof (struct wally_psbt_input *input)
```

Clear the explicit asset surjection proof in an input.

Parameters

- **input** – The input to update.

Returns See *Error Codes*

```
int wally_psbt_input_set_issuance_amount (struct wally_psbt_input *input, uint64_t amount)
```

Set the unblinded token issuance amount in an input.

Parameters

- **input** – The input to update.

- **amount** – The issuance amount.

Note: Setting the amount to zero indicates no issuance.

Returns See [Error Codes](#)

int **wally_psbt_input_set_inflation_keys** (struct wally_psbt_input *input, uint64_t value)
Set the unblinded number of inflation (reissuance) keys in an input.

Parameters

- **input** – The input to update.
- **value** – The number of inflation keys.

Returns See [Error Codes](#)

int **wally_psbt_input_set_pegin_amount** (struct wally_psbt_input *input, uint64_t amount)
Set the peg-in amount in an input.

Parameters

- **input** – The input to update.
- **amount** – The peg-in amount.

Returns See [Error Codes](#)

int **wally_psbt_input_set_pegin_tx** (struct wally_psbt_input *input, const struct wally_tx *tx)
Set the peg-in transaction in an input.

Parameters

- **input** – The input to update.
- **tx** – The (non witness) peg-in transaction for this input if it exists.

Returns See [Error Codes](#)

int **wally_psbt_input_set_pegin_witness** (struct wally_psbt_input *input, const struct wally_tx_witness_stack *witness)
Set the peg-in witness in an input.

Parameters

- **input** – The input to update.
- **witness** – The peg-in witness stack for the input, or NULL if not present.

Returns See [Error Codes](#)

int **wally_psbt_input_get_pegin_txout_proof** (const struct wally_psbt_input *input, unsigned char *bytes_out, size_t len, size_t *written)
Get the peg-in transaction output proof from an input.

Parameters

- **input** – The input to get from.
- **bytes_out** – Destination for the peg-in transaction output proof.
- **len** – Size of bytes_out in bytes.
- **written** – Destination for the number of bytes written to bytes_out. Will be zero if the value is not present.

Note: This operates on the PSET field `PSBT_ELEMENTS_IN_PEG_IN_TXOUT_PROOF`.

Returns See *Variable Length Output Buffers*

int **wally_psbt_input_get_pegin_txout_proof_len** (const struct wally_psbt_input *input,
size_t *written)
Get the length of a peg-in transaction output proof from an input.

Parameters

- **input** – The input to get from.
- **written** – Destination for the length, or zero if not present.

Returns See *Error Codes*

int **wally_psbt_input_set_pegin_txout_proof** (struct wally_psbt_input *input, const unsigned
char *txout_proof, size_t txout_proof_len)
Set the peg-in transaction output proof in an input.

Parameters

- **input** – The input to update.
- **txout_proof** – The peg-in transaction output proof.
- **txout_proof_len** – Size of txout_proof in bytes.

Returns See *Error Codes*

int **wally_psbt_input_clear_pegin_txout_proof** (struct wally_psbt_input *input)
Clear the peg-in transaction output proof in an input.

Parameters

- **input** – The input to update.

Returns See *Error Codes*

int **wally_psbt_input_get_pegin_genesis_blockhash** (const struct wally_psbt_input *input,
unsigned char *bytes_out, size_t len,
size_t *written)
Get the peg-in genesis blockhash from an input.

Parameters

- **input** – The input to get from.
- **bytes_out** – Destination for the peg-in genesis blockhash.
- **len** – Size of bytes_out in bytes.
- **written** – Destination for the number of bytes written to bytes_out. Will be zero if the value is not present.

Returns See *Variable Length Output Buffers*

int **wally_psbt_input_get_pegin_genesis_blockhash_len** (const struct wally_psbt_input *in-
put, size_t *written)
Get the length of a peg-in genesis blockhash from an input.

Parameters

- **input** – The input to get from.

- **written** – Destination for the length, or zero if not present.

Note: This operates on the PSET field `PSBT_ELEMENTS_IN_PEG_IN_GENESIS_HASH`.

Returns See *Error Codes*

```
int wally_psbt_input_set_pegin_genesis_blockhash (struct wally_psbt_input *input, const
                                                    unsigned char *genesis_blockhash,
                                                    size_t genesis_blockhash_len)
```

Set the peg-in genesis blockhash in an input.

Parameters

- **input** – The input to update.
- **genesis_blockhash** – The peg-in genesis blockhash.
- **genesis_blockhash_len** – Size of `genesis_blockhash` in bytes. Must be `WALLY_TXHASH_LEN`.

Returns See *Error Codes*

```
int wally_psbt_input_clear_pegin_genesis_blockhash (struct wally_psbt_input *input)
Clear the peg-in genesis blockhash in an input.
```

Parameters

- **input** – The input to update.

Returns See *Error Codes*

```
int wally_psbt_input_get_pegin_claim_script (const struct wally_psbt_input *input, unsigned
                                              char *bytes_out, size_t len, size_t *written)
```

Get the peg-in claim script from an input.

Parameters

- **input** – The input to get from.
- **bytes_out** – Destination for the peg-in claim script.
- **len** – Size of `bytes_out` in bytes.
- **written** – Destination for the number of bytes written to `bytes_out`. Will be zero if the value is not present.

Note: This operates on the PSET field `PSBT_ELEMENTS_IN_PEG_IN_CLAIM_SCRIPT`.

Returns See *Variable Length Output Buffers*

```
int wally_psbt_input_get_pegin_claim_script_len (const struct wally_psbt_input *input,
                                                  size_t *written)
```

Get the length of a peg-in claim script from an input.

Parameters

- **input** – The input to get from.
- **written** – Destination for the length, or zero if not present.

Returns See *Error Codes*

int **wally_psbt_input_set_pegin_claim_script** (struct wally_psbt_input *input, const unsigned char *script, size_t script_len)

Set the peg-in claim script in an input.

Parameters

- **input** – The input to update.
- **script** – The peg-in claim script.
- **script_len** – Size of script in bytes.

Returns See [Error Codes](#)

int **wally_psbt_input_clear_pegin_claim_script** (struct wally_psbt_input *input)

Clear the peg-in claim script in an input.

Parameters

- **input** – The input to update.

Returns See [Error Codes](#)

int **wally_psbt_input_get_issuance_amount_commitment** (const struct wally_psbt_input *input, unsigned char *bytes_out, size_t len, size_t *written)

Get the blinded token issuance amount from an input.

Parameters

- **input** – The input to get from.
- **bytes_out** – Destination for the blinded issuance amount.
- **len** – Size of bytes_out in bytes. Must be ASSET_COMMITMENT_LEN.
- **written** – Destination for the number of bytes written to bytes_out. Will be zero if the value is not present.

Note: This operates on the PSET field PSBT_ELEMENTS_IN_ISSUANCE_VALUE_COMMITMENT.

Returns See [Variable Length Output Buffers](#)

int **wally_psbt_input_get_issuance_amount_commitment_len** (const struct wally_psbt_input *input, size_t *written)

Get the length of a blinded token issuance amount from an input.

Parameters

- **input** – The input to get from.
- **written** – Destination for the length, or zero if not present.

Returns See [Error Codes](#)

int **wally_psbt_input_set_issuance_amount_commitment** (struct wally_psbt_input *input, const unsigned char *commitment, size_t commitment_len)

Set the blinded token issuance amount in an input.

Parameters

- **input** – The input to update.

- **commitment** – The blinded issuance amount commitment.
- **commitment_len** – Size of commitment in bytes. Must be ASSET_COMMITMENT_LEN.

Returns See *Error Codes*

int **wally_psbt_input_clear_issuance_amount_commitment** (struct wally_psbt_input *input)
Clear the blinded token issuance amount in an input.

Parameters

- **input** – The input to update.

Returns See *Error Codes*

int **wally_psbt_input_get_issuance_amount_rangeproof** (const struct wally_psbt_input *input, unsigned char *bytes_out, size_t len, size_t *written)

Get the issuance amount rangeproof from an input.

Parameters

- **input** – The input to get from.
- **bytes_out** – Destination for the issuance amount rangeproof.
- **len** – Size of bytes_out in bytes.
- **written** – Destination for the number of bytes written to bytes_out. Will be zero if the value is not present.

Note: This operates on the PSET field PSBT_ELEMENTS_IN_ISSUANCE_VALUE_RANGEPROOF.

Returns See *Variable Length Output Buffers*

int **wally_psbt_input_get_issuance_amount_rangeproof_len** (const struct wally_psbt_input *input, size_t *written)

Get the length of the issuance amount rangeproof from an input.

Parameters

- **input** – The input to get from.
- **written** – Destination for the length, or zero if not present.

Returns See *Error Codes*

int **wally_psbt_input_set_issuance_amount_rangeproof** (struct wally_psbt_input *input, const unsigned char *rangeproof, size_t rangeproof_len)

Set the issuance amount rangeproof in an input.

Parameters

- **input** – The input to update.
- **rangeproof** – The issuance amount rangeproof.
- **rangeproof_len** – Size of rangeproof in bytes.

Returns See *Error Codes*

int **wally_psbt_input_clear_issuance_amount_rangeproof** (struct wally_psbt_input *input)
Clear the issuance amount rangeproof in an input.

Parameters

- **input** – The input to update.

Returns See *Error Codes*

int **wally_psbt_input_get_issuance_blinding_nonce** (const struct wally_psbt_input *input,
unsigned char *bytes_out, size_t len,
size_t *written)

Get the asset issuance blinding nonce from an input.

Parameters

- **input** – The input to get from.
- **bytes_out** – Destination for the asset issuance blinding nonce.
- **len** – Size of bytes_out in bytes. Must be BLINDING_FACTOR_LEN.
- **written** – Destination for the number of bytes written to bytes_out. Will be zero if the value is not present.

Note: This operates on the PSET field PSBT_ELEMENTS_IN_ISSUANCE_BLINDING_NONCE.

Returns See *Variable Length Output Buffers*

int **wally_psbt_input_get_issuance_blinding_nonce_len** (const struct wally_psbt_input *input,
size_t *written)

Get the length of a asset issuance blinding nonce from an input.

Parameters

- **input** – The input to get from.
- **written** – Destination for the length, or zero if not present.

Returns See *Error Codes*

int **wally_psbt_input_set_issuance_blinding_nonce** (struct wally_psbt_input *input,
const unsigned char *nonce,
size_t nonce_len)

Set the asset issuance blinding nonce in an input.

Parameters

- **input** – The input to update.
- **nonce** – Asset issuance or revelation blinding nonce.
- **nonce_len** – Size of nonce in bytes. Must be ASSET_TAG_LEN.

Returns See *Error Codes*

int **wally_psbt_input_clear_issuance_blinding_nonce** (struct wally_psbt_input *input)
Clear the asset issuance blinding nonce in an input.

Parameters

- **input** – The input to update.

Returns See *Error Codes*

```
int wally_psbtx_input_get_issuance_asset_entropy (const struct wally_psbtx_input *input,
                                                  unsigned char *bytes_out, size_t len,
                                                  size_t *written)
```

Get the asset issuance entropy from an input.

Parameters

- **input** – The input to get from.
- **bytes_out** – Destination for the asset issuance entropy.
- **len** – Size of `bytes_out` in bytes. Must be `BLINDING_FACTOR_LEN`.
- **written** – Destination for the number of bytes written to `bytes_out`. Will be zero if the value is not present.

Note: This operates on the PSET field `PSBT_ELEMENTS_IN_ISSUANCE_ASSET_ENTROPY`.

Returns See *Variable Length Output Buffers*

```
int wally_psbtx_input_get_issuance_asset_entropy_len (const struct wally_psbtx_input *input,
                                                      size_t *written)
```

Get the length of a asset issuance entropy from an input.

Parameters

- **input** – The input to get from.
- **written** – Destination for the length, or zero if not present.

Returns See *Error Codes*

```
int wally_psbtx_input_set_issuance_asset_entropy (struct wally_psbtx_input *input, const unsigned char *entropy,
                                                  size_t entropy_len)
```

Set the asset issuance entropy in an input.

Parameters

- **input** – The input to update.
- **entropy** – The asset issuance entropy.
- **entropy_len** – Size of `entropy` in bytes.

Returns See *Error Codes*

```
int wally_psbtx_input_clear_issuance_asset_entropy (struct wally_psbtx_input *input)
```

Clear the asset issuance entropy in an input.

Parameters

- **input** – The input to update.

Returns See *Error Codes*

```
int wally_psbtx_input_get_issuance_amount_blinding_rangeproof (const struct wally_psbtx_input *input,
                                                                unsigned char *bytes_out,
                                                                size_t len,
                                                                size_t *written)
```

Get the issuance amount blinding rangeproof from an input.

Parameters

- **input** – The input to get from.
- **bytes_out** – Destination for the issuance amount blinding rangeproof.
- **len** – Size of `bytes_out` in bytes.
- **written** – Destination for the number of bytes written to `bytes_out`. Will be zero if the value is not present.

Note: This operates on the PSET field `PSBT_ELEMENTS_IN_ISSUANCE_BLIND_VALUE_PROOF`.

Returns See *Variable Length Output Buffers*

```
int wally_psb_t_input_get_issuance_amount_blinding_rangeproof_len (const struct wally_psb_t_input *input, size_t *written)
```

Get the length of a issuance amount blinding rangeproof from an input.

Parameters

- **input** – The input to get from.
- **written** – Destination for the length, or zero if not present.

Returns See *Error Codes*

```
int wally_psb_t_input_set_issuance_amount_blinding_rangeproof (struct wally_psb_t_input *input, const unsigned char *rangeproof, size_t rangeproof_len)
```

Set the issuance amount blinding rangeproof in an input.

Parameters

- **input** – The input to update.
- **rangeproof** – The issuance amount blinding rangeproof.
- **rangeproof_len** – Size of `rangeproof` in bytes.

Returns See *Error Codes*

```
int wally_psb_t_input_clear_issuance_amount_blinding_rangeproof (struct wally_psb_t_input *input)
```

Clear the issuance amount blinding rangeproof in an input.

Parameters

- **input** – The input to update.

Returns See *Error Codes*

```
int wally_psb_t_input_get_inflation_keys_commitment (const struct wally_psb_t_input *input, unsigned char *bytes_out, size_t len, size_t *written)
```

Get the blinded number of reissuance tokens from an input.

Parameters

- **input** – The input to get from.
- **bytes_out** – Destination for the blinded number of reissuance tokens.
- **len** – Size of **bytes_out** in bytes. Must be `ASSET_COMMITMENT_LEN`.
- **written** – Destination for the number of bytes written to **bytes_out**. Will be zero if the value is not present.

Note: This operates on the PSET field `PSBT_ELEMENTS_IN_INFLATION_KEYS_COMMITMENT`.

Returns See *Variable Length Output Buffers*

```
int wally_psbt_input_get_inflation_keys_commitment_len (const struct wally_psbt_input *input, size_t *written)
```

Get the length of the blinded number of reissuance tokens from an input.

Parameters

- **input** – The input to get from.
- **written** – Destination for the length, or zero if not present.

Returns See *Error Codes*

```
int wally_psbt_input_set_inflation_keys_commitment (struct wally_psbt_input *input, const unsigned char *commitment, size_t commitment_len)
```

Set the blinded number of reissuance tokens in an input.

Parameters

- **input** – The input to update.
- **commitment** – The blinded number of reissuance tokens.
- **commitment_len** – Size of commitment in bytes. Must be `ASSET_COMMITMENT_LEN`.

Returns See *Error Codes*

```
int wally_psbt_input_clear_inflation_keys_commitment (struct wally_psbt_input *input)
```

Clear the blinded number of reissuance tokens in an input.

Parameters

- **input** – The input to update.

Returns See *Error Codes*

```
int wally_psbt_input_get_inflation_keys_rangeproof (const struct wally_psbt_input *input, unsigned char *bytes_out, size_t len, size_t *written)
```

Get the reissuance tokens rangeproof from an input.

Parameters

- **input** – The input to get from.
- **bytes_out** – Destination for the reissuance tokens rangeproof.
- **len** – Size of **bytes_out** in bytes.

- **written** – Destination for the number of bytes written to `bytes_out`. Will be zero if the value is not present.

Note: This operates on the PSET field `PSBT_ELEMENTS_IN_INFLATION_KEYS_RANGEPROOF`.

Returns See *Variable Length Output Buffers*

```
int wally_psbt_input_get_inflation_keys_rangeproof_len (const      struct
                                                         wally_psbt_input  *input,
                                                         size_t *written)
```

Get the length of a reissuance tokens rangeproof from an input.

Parameters

- **input** – The input to get from.
- **written** – Destination for the length, or zero if not present.

Returns See *Error Codes*

```
int wally_psbt_input_set_inflation_keys_rangeproof (struct wally_psbt_input *input,
                                                         const unsigned char *rangeproof,
                                                         size_t rangeproof_len)
```

Set the reissuance tokens rangeproof in an input.

Parameters

- **input** – The input to update.
- **rangeproof** – The reissuance tokens rangeproof.
- **rangeproof_len** – Size of `rangeproof` in bytes.

Returns See *Error Codes*

```
int wally_psbt_input_clear_inflation_keys_rangeproof (struct wally_psbt_input *input)
```

Clear the reissuance tokens rangeproof in an input.

Parameters

- **input** – The input to update.

Returns See *Error Codes*

```
int wally_psbt_input_get_inflation_keys_blinding_rangeproof (const      struct
                                                             wally_psbt_input  *input,
                                                             unsigned
                                                             char              *bytes_out,
                                                             size_t len, size_t *written)
```

Get the reissuance tokens blinding rangeproof from an input.

Parameters

- **input** – The input to get from.
- **bytes_out** – Destination for the reissuance tokens blinding rangeproof.
- **len** – Size of `bytes_out` in bytes.
- **written** – Destination for the number of bytes written to `bytes_out`. Will be zero if the value is not present.

Note: This operates on the PSET field PSBT_ELEMENTS_IN_ISSUANCE_BLIND_INFLATION_KEYS_PROOF.

Returns See *Variable Length Output Buffers*

```
int wally_psbt_input_get_inflation_keys_blinding_rangeproof_len (const      struct
                                                                wally_psbt_input *in-
                                                                put, size_t *writ-
                                                                ten)
```

Get the length of a reissuance tokens blinding rangeproof from an input.

Parameters

- **input** – The input to get from.
- **written** – Destination for the length, or zero if not present.

Returns See *Error Codes*

```
int wally_psbt_input_set_inflation_keys_blinding_rangeproof (struct
                                                                wally_psbt_input  *in-
                                                                put, const unsigned
                                                                char      *rangeproof,
                                                                size_t rangeproof_len)
```

Set the reissuance tokens blinding rangeproof in an input.

Parameters

- **input** – The input to update.
- **rangeproof** – The reissuance tokens blinding rangeproof.
- **rangeproof_len** – Size of rangeproof in bytes.

Returns See *Error Codes*

```
int wally_psbt_input_clear_inflation_keys_blinding_rangeproof (struct
                                                                wally_psbt_input *in-
                                                                put)
```

Clear the reissuance tokens blinding rangeproof in an input.

Parameters

- **input** – The input to update.

Returns See *Error Codes*

```
int wally_psbt_input_get_utxo_rangeproof (const struct wally_psbt_input *input, unsigned
                                                                char *bytes_out, size_t len, size_t *written)
```

Get the UTXO rangeproof from an input.

Parameters

- **input** – The input to get from.
- **bytes_out** – Destination for the UTXO rangeproof.
- **len** – Size of bytes_out in bytes.
- **written** – Destination for the number of bytes written to bytes_out. Will be zero if the value is not present.

Note: This operates on the PSET field `PSBT_ELEMENTS_IN_UTXO_RANGEPROOF`.

Returns See [Variable Length Output Buffers](#)

```
int wally_psbt_input_get_utxo_rangeproof_len (const struct wally_psbt_input *input,
                                              size_t *written)
```

Get the length of a UTXO rangeproof from an input.

Parameters

- **input** – The input to get from.
- **written** – Destination for the length, or zero if not present.

Returns See [Error Codes](#)

```
int wally_psbt_input_set_utxo_rangeproof (struct wally_psbt_input *input, const unsigned
                                          char *rangeproof, size_t rangeproof_len)
```

Set the UTXO rangeproof in an input.

Parameters

- **input** – The input to update.
- **rangeproof** – The UTXO rangeproof.
- **rangeproof_len** – Size of rangeproof in bytes.

Returns See [Error Codes](#)

```
int wally_psbt_input_clear_utxo_rangeproof (struct wally_psbt_input *input)
```

Clear the UTXO rangeproof in an input.

Parameters

- **input** – The input to update.

Returns See [Error Codes](#)

```
int wally_psbt_input_generate_explicit_proofs (struct wally_psbt_input *input,
                                                uint64_t satoshi, const unsigned char *asset,
                                                size_t asset_len, const unsigned char *abf,
                                                size_t abf_len, const unsigned char *vbf,
                                                size_t vbf_len, const unsigned char *entropy,
                                                size_t entropy_len)
```

Generate explicit proofs and unblinded values from an inputs witness UTXO.

Parameters

- **input** – The input to generate proofs for.
- **satoshi** – The explicit value of the input.
- **asset** – The explicit asset tag.
- **asset_len** – Size of asset in bytes. Must be `ASSET_TAG_LEN`.
- **abf** – Asset blinding factor.
- **abf_len** – Length of abf. Must be `BLINDING_FACTOR_LEN`.
- **vbf** – Value blinding factor.
- **vbf_len** – Length of vbf. Must be `BLINDING_FACTOR_LEN`.

- **entropy** – Random entropy for explicit range proof generation.
- **entropy_len** – Size of `entropy` in bytes. Must be `BLINDING_FACTOR_LEN`.

Note: This function exposes the unblinded asset and value in the PSET,

which is only appropriate in certain multi-party protocols.

Returns See *Error Codes*

int **wally_psbt_input_is_finalized** (const struct wally_psbt_input *input, size_t *written)
Determine if a PSBT input is finalized.

Parameters

- **input** – The input to check.
- **written** – On success, set to one if the input is finalized, otherwise zero.

Returns See *Error Codes*

int **wally_psbt_output_set_redeem_script** (struct wally_psbt_output *output, const unsigned char *script, size_t script_len)

Set the redeem_script in an output.

Parameters

- **output** – The input to update.
- **script** – The redeem script for this output.
- **script_len** – Length of `script` in bytes.

Returns See *Error Codes*

int **wally_psbt_output_set_witness_script** (struct wally_psbt_output *output, const unsigned char *script, size_t script_len)

Set the witness_script in an output.

Parameters

- **output** – The output to update.
- **script** – The witness script for this output.
- **script_len** – Length of `script` in bytes.

Returns See *Error Codes*

int **wally_psbt_output_set_keypaths** (struct wally_psbt_output *output, const struct wally_map *map_in)

Set the keypaths in an output.

Parameters

- **output** – The output to update.
- **map_in** – The HD keypaths for this output.

Returns See *Error Codes*

int **wally_psbt_output_find_keypath** (struct wally_psbt_output *output, const unsigned char *pub_key, size_t pub_key_len, size_t *written)

Find a keypath matching a pubkey in an output.

Parameters

- **output** – The output to search in.
- **pub_key** – The pubkey to find.
- **pub_key_len** – Length of `pub_key` in bytes. Must be `EC_PUBLIC_KEY_UNCOMPRESSED_LEN` or `EC_PUBLIC_KEY_LEN`.
- **written** – On success, set to zero if the item is not found, otherwise the index of the item plus one.

Returns See *Error Codes*

```
int wally_psbt_output_keypath_add (struct wally_psbt_output *output, const unsigned
                                   char *pub_key, size_t pub_key_len, const unsigned char *fin-
                                   gerprint, size_t fingerprint_len, const uint32_t *child_path,
                                   size_t child_path_len)
```

Convert and add a pubkey/keypath to an output.

Parameters

- **output** – The output to add to.
- **pub_key** – The pubkey to add.
- **pub_key_len** – Length of `pub_key` in bytes. Must be `EC_PUBLIC_KEY_UNCOMPRESSED_LEN` or `EC_PUBLIC_KEY_LEN`.
- **fingerprint** – The master key fingerprint for the pubkey.
- **fingerprint_len** – Length of fingerprint in bytes. Must be `BIP32_KEY_FINGERPRINT_LEN`.
- **child_path** – The BIP32 derivation path for the pubkey.
- **child_path_len** – The number of items in `child_path`.

Returns See *Error Codes*

```
int wally_psbt_output_set_unknowns (struct wally_psbt_output *output, const struct
                                   wally_map *map_in)
```

Set the unknown map in an output.

Parameters

- **output** – The output to update.
- **map_in** – The unknown key value pairs for this output.

Returns See *Error Codes*

```
int wally_psbt_output_find_unknown (struct wally_psbt_output *output, const unsigned char *key,
                                    size_t key_len, size_t *written)
```

Find an unknown item matching a key in an output.

Parameters

- **output** – The output to search in.
- **key** – The key to find.
- **key_len** – Length of `key` in bytes.
- **written** – On success, set to zero if the item is not found, otherwise the index of the item plus one.

Returns See *Error Codes*

int **wally_psbt_output_set_amount** (struct wally_psbt_output **output*, uint64_t *amount*)
Set the amount in an output.

Parameters

- **output** – The output to update.
- **amount** – The amount for this output.

Returns See *Error Codes*

int **wally_psbt_output_clear_amount** (struct wally_psbt_output **output*)
Clear the amount in an output.

Parameters

- **output** – The output to update.

Returns See *Error Codes*

int **wally_psbt_output_set_script** (struct wally_psbt_output **output*, const unsigned char **script*, size_t *script_len*)
Set the script in an output.

Parameters

- **output** – The output to update.
- **script** – The script for this output.
- **script_len** – Length of *script* in bytes.

Returns See *Error Codes*

int **wally_psbt_output_set_blinder_index** (struct wally_psbt_output **output*, uint32_t *index*)
Set the input blinder index in an output.

Parameters

- **output** – The output to update.
- **index** – The input blinder index for this output.

Returns See *Error Codes*

int **wally_psbt_output_clear_blinder_index** (struct wally_psbt_output **output*)
Clear the input blinder index from an output.

Parameters

- **output** – The output to update.

Returns See *Error Codes*

int **wally_psbt_output_get_value_commitment** (const struct wally_psbt_output **output*, unsigned char **bytes_out*, size_t *len*, size_t **written*)
Get the blinded asset value from an output.

Parameters

- **output** – The output to get from.
- **bytes_out** – Destination for the blinded asset value.
- **len** – Size of *bytes_out* in bytes.
- **written** – Destination for the number of bytes written to *bytes_out*. Will be zero if the value is not present.

Note: This operates on the PSET field `PSBT_ELEMENTS_OUT_VALUE_COMMITMENT`.

Returns See *Variable Length Output Buffers*

int **wally_psbt_output_get_value_commitment_len** (const struct wally_psbt_output *output,
size_t *written)

Get the length of the blinded asset value from an output.

Parameters

- **output** – The output to get from.
- **written** – Destination for the length, or zero if not present.

Returns See *Error Codes*

int **wally_psbt_output_set_value_commitment** (struct wally_psbt_output *output, const unsigned
char *commitment, size_t commitment_len)

Set the blinded asset value in an output.

Parameters

- **output** – The output to update.
- **commitment** – The blinded asset value.
- **commitment_len** – Size of commitment in bytes.

Returns See *Error Codes*

int **wally_psbt_output_clear_value_commitment** (struct wally_psbt_output *output)

Clear the blinded asset value in an output.

Parameters

- **output** – The output to update.

Returns See *Error Codes*

int **wally_psbt_output_get_asset** (const struct wally_psbt_output *output, unsigned char *bytes_out,
size_t len, size_t *written)

Get the asset tag from an output.

Parameters

- **output** – The output to get from.
- **bytes_out** – Destination for the asset tag.
- **len** – Size of bytes_out in bytes.
- **written** – Destination for the number of bytes written to bytes_out. Will be zero if the value is not present.

Note: This operates on the PSET field `PSBT_ELEMENTS_OUT_ASSET`.

Returns See *Variable Length Output Buffers*

int **wally_psbt_output_get_asset_len** (const struct wally_psbt_output *output, size_t *written)

Get the length of the asset tag from an output.

Parameters

- **output** – The output to get from.
- **written** – Destination for the length, or zero if not present.

Returns See *Error Codes*

int **wally_psbtx_output_set_asset** (struct wally_psbtx_output **output*, const unsigned char **asset*, size_t *asset_len*)

Set the asset tag in an output.

Parameters

- **output** – The output to update.
- **asset** – The asset tag.
- **asset_len** – Size of *asset* in bytes. Must be ASSET_TAG_LEN.

Returns See *Error Codes*

int **wally_psbtx_output_clear_asset** (struct wally_psbtx_output **output*)

Clear the asset tag in an output.

Parameters

- **output** – The output to update.

Returns See *Error Codes*

int **wally_psbtx_output_get_asset_commitment** (const struct wally_psbtx_output **output*, unsigned char **bytes_out*, size_t *len*, size_t **written*)

Get the blinded asset tag from an output.

Parameters

- **output** – The output to get from.
- **bytes_out** – Destination for the blinded asset tag.
- **len** – Size of *bytes_out* in bytes.
- **written** – Destination for the number of bytes written to *bytes_out*. Will be zero if the value is not present.

Note: This operates on the PSET field PSBT_ELEMENTS_OUT_ASSET_COMMITMENT.

Returns See *Variable Length Output Buffers*

int **wally_psbtx_output_get_asset_commitment_len** (const struct wally_psbtx_output **output*, size_t **written*)

Get the length of the blinded asset tag from an output.

Parameters

- **output** – The output to get from.
- **written** – Destination for the length, or zero if not present.

Returns See *Error Codes*

int **wally_psbtx_output_set_asset_commitment** (struct wally_psbtx_output **output*, const unsigned char **commitment*, size_t *commitment_len*)

Set the blinded asset tag in an output.

Parameters

- **output** – The output to update.
- **commitment** – The blinded asset tag.
- **commitment_len** – Size of `commitment` in bytes.

Returns See *Error Codes*

int **wally_psbt_output_clear_asset_commitment** (struct wally_psbt_output **output*)
Clear the blinded asset tag in an output.

Parameters

- **output** – The output to update.

Returns See *Error Codes*

int **wally_psbt_output_get_value_rangeproof** (const struct wally_psbt_output **output*, unsigned char **bytes_out*, size_t *len*, size_t **written*)
Get the output value range proof from an output.

Parameters

- **output** – The output to get from.
- **bytes_out** – Destination for the output value range proof.
- **len** – Size of `bytes_out` in bytes.
- **written** – Destination for the number of bytes written to `bytes_out`. Will be zero if the value is not present.

Note: This operates on the PSET field `PSBT_ELEMENTS_OUT_VALUE_RANGEPROOF`.

Returns See *Variable Length Output Buffers*

int **wally_psbt_output_get_value_rangeproof_len** (const struct wally_psbt_output **output*, size_t **written*)
Get the length of the output value range proof from an output.

Parameters

- **output** – The output to get from.
- **written** – Destination for the length, or zero if not present.

Returns See *Error Codes*

int **wally_psbt_output_set_value_rangeproof** (struct wally_psbt_output **output*, const unsigned char **rangeproof*, size_t *rangeproof_len*)
Set the output value range proof in an output.

Parameters

- **output** – The output to update.
- **rangeproof** – The output value range proof.
- **rangeproof_len** – Size of `rangeproof` in bytes.

Returns See *Error Codes*

int **wally_psbt_output_clear_value_rangeproof** (struct wally_psbt_output **output*)
Clear the output value range proof in an output.

Parameters

- **output** – The output to update.

Returns See *Error Codes*

```
int wally_psbtx_output_get_asset_surjectionproof (const struct wally_psbtx_output *output,
                                                  unsigned char *bytes_out, size_t len,
                                                  size_t *written)
```

Get the asset surjection proof from an output.

Parameters

- **output** – The output to get from.
- **bytes_out** – Destination for the asset surjection proof.
- **len** – Size of `bytes_out` in bytes.
- **written** – Destination for the number of bytes written to `bytes_out`. Will be zero if the value is not present.

Note: This operates on the PSET field `PSBT_ELEMENTS_OUT_ASSET_SURJECTION_PROOF`.

Returns See *Variable Length Output Buffers*

```
int wally_psbtx_output_get_asset_surjectionproof_len (const struct wally_psbtx_output *output,
                                                      size_t *written)
```

Get the length of the asset surjection proof from an output.

Parameters

- **output** – The output to get from.
- **written** – Destination for the length, or zero if not present.

Returns See *Error Codes*

```
int wally_psbtx_output_set_asset_surjectionproof (struct wally_psbtx_output *output,
                                                  const unsigned char *surjectionproof,
                                                  size_t surjectionproof_len)
```

Set the asset surjection proof in an output.

Parameters

- **output** – The output to update.
- **surjectionproof** – The asset surjection proof.
- **surjectionproof_len** – Size of `surjectionproof` in bytes.

Returns See *Error Codes*

```
int wally_psbtx_output_clear_asset_surjectionproof (struct wally_psbtx_output *output)
```

Clear the asset surjection proof in an output.

Parameters

- **output** – The output to update.

Returns See *Error Codes*

```
int wally_psbt_output_get_blinding_public_key (const struct wally_psbt_output *output,
                                              unsigned char *bytes_out, size_t len,
                                              size_t *written)
```

Get the blinding public key from an output.

Parameters

- **output** – The output to get from.
- **bytes_out** – Destination for the blinding public key.
- **len** – Size of `bytes_out` in bytes.
- **written** – Destination for the number of bytes written to `bytes_out`. Will be zero if the value is not present.

Note: This operates on the PSET field `PSBT_ELEMENTS_OUT_BLINDING_PUBKEY`.

Returns See *Variable Length Output Buffers*

```
int wally_psbt_output_get_blinding_public_key_len (const struct wally_psbt_output *out-
                                                  put, size_t *written)
```

Get the length of the blinding public key from an output.

Parameters

- **output** – The output to get from.
- **written** – Destination for the length, or zero if not present.

Returns See *Error Codes*

```
int wally_psbt_output_set_blinding_public_key (struct wally_psbt_output *output, const un-
                                              signed char *pub_key, size_t pub_key_len)
```

Set the blinding public key in an output.

Parameters

- **output** – The output to update.
- **pub_key** – The blinding public key.
- **pub_key_len** – Size of `pub_key` in bytes.

Returns See *Error Codes*

```
int wally_psbt_output_clear_blinding_public_key (struct wally_psbt_output *output)
```

Clear the blinding public key in an output.

Parameters

- **output** – The output to update.

Returns See *Error Codes*

```
int wally_psbt_output_get_ecdh_public_key (const struct wally_psbt_output *output, unsigned
                                          char *bytes_out, size_t len, size_t *written)
```

Get the ephemeral ECDH public key from an output.

Parameters

- **output** – The output to get from.
- **bytes_out** – Destination for the ephemeral ECDH public key.

- **len** – Size of `bytes_out` in bytes.
- **written** – Destination for the number of bytes written to `bytes_out`. Will be zero if the value is not present.

Note: This operates on the PSET field `PSBT_ELEMENTS_OUT_ECDH_PUBKEY`.

Returns See *Variable Length Output Buffers*

int **wally_psbt_output_get_ecdh_public_key_len** (const struct wally_psbt_output *output,
size_t *written)
Get the length of the ephemeral ECDH public key from an output.

Parameters

- **output** – The output to get from.
- **written** – Destination for the length, or zero if not present.

Returns See *Error Codes*

int **wally_psbt_output_set_ecdh_public_key** (struct wally_psbt_output *output, const unsigned
char *pub_key, size_t pub_key_len)
Set the ephemeral ECDH public key in an output.

Parameters

- **output** – The output to update.
- **pub_key** – The ephemeral ECDH public key.
- **pub_key_len** – Size of `pub_key` in bytes.

Returns See *Error Codes*

int **wally_psbt_output_clear_ecdh_public_key** (struct wally_psbt_output *output)
Clear the ephemeral ECDH public key in an output.

Parameters

- **output** – The output to update.

Returns See *Error Codes*

int **wally_psbt_output_get_value_blinding_rangeproof** (const struct
wally_psbt_output *output, un-
signed char *bytes_out, size_t len,
size_t *written)

Get the asset value blinding rangeproof from an output.

Parameters

- **output** – The output to get from.
- **bytes_out** – Destination for the asset value blinding rangeproof.
- **len** – Size of `bytes_out` in bytes.
- **written** – Destination for the number of bytes written to `bytes_out`. Will be zero if the value is not present.

Note: This operates on the PSET field `PSBT_ELEMENTS_OUT_BLIND_VALUE_PROOF`.

Returns See *Variable Length Output Buffers*

```
int wally_psbt_output_get_value_blinding_rangeproof_len (const struct wally_psbt_output *output, size_t *written)
```

Get the length of the asset value blinding rangeproof from an output.

Parameters

- **output** – The output to get from.
- **written** – Destination for the length, or zero if not present.

Returns See *Error Codes*

```
int wally_psbt_output_set_value_blinding_rangeproof (struct wally_psbt_output *output, const unsigned char *rangeproof, size_t rangeproof_len)
```

Set the asset value blinding rangeproof in an output.

Parameters

- **output** – The output to update.
- **rangeproof** – The asset value blinding rangeproof.
- **rangeproof_len** – Size of rangeproof in bytes.

Returns See *Error Codes*

```
int wally_psbt_output_clear_value_blinding_rangeproof (struct wally_psbt_output *output)
```

Clear the asset value blinding rangeproof in an output.

Parameters

- **output** – The output to update.

Returns See *Error Codes*

```
int wally_psbt_output_get_asset_blinding_surjectionproof (const struct wally_psbt_output *output, unsigned char *bytes_out, size_t len, size_t *written)
```

Get the asset tag blinding surjection proof from an output.

Parameters

- **output** – The output to get from.
- **bytes_out** – Destination for the asset tag blinding surjection proof.
- **len** – Size of bytes_out in bytes.
- **written** – Destination for the number of bytes written to bytes_out. Will be zero if the value is not present.

Note: This operates on the PSET field PSBT_ELEMENTS_OUT_BLIND_ASSET_PROOF.

Returns See *Variable Length Output Buffers*

```
int wally_psbt_output_get_asset_blinding_surjectionproof_len (const struct wally_psbt_output *output, size_t *written)
```

Get the length of the asset tag blinding surjection proof from an output.

Parameters

- **output** – The output to get from.
- **written** – Destination for the length, or zero if not present.

Returns See *Error Codes*

```
int wally_psbt_output_set_asset_blinding_surjectionproof (struct wally_psbt_output *output, const unsigned char *surjectionproof, size_t surjectionproof_len)
```

Set the asset tag blinding surjection proof in an output.

Parameters

- **output** – The output to update.
- **surjectionproof** – The asset tag blinding surjection proof.
- **surjectionproof_len** – Size of `surjectionproof` in bytes.

Returns See *Error Codes*

```
int wally_psbt_output_clear_asset_blinding_surjectionproof (struct wally_psbt_output *output)
```

Clear the asset tag blinding surjection proof in an output.

Parameters

- **output** – The output to update.

Returns See *Error Codes*

```
int wally_psbt_output_get_blinding_status (const struct wally_psbt_output *output, uint32_t *flags, size_t *written)
```

Get the blinding status of an output.

Parameters

- **output** – The output to get the blinding status from.
- **flags** – Flags controlling the checks to perform. Must be 0.
- **written** – Destination for the blinding status: `WALLY_PSET_BLIINDED_NONE` if unblinded, `WALLY_PSET_BLIINDED_REQUIRED` if only the blinding public key is present, `WALLY_PSET_BLIINDED_FULL` or `WALLY_PSET_BLIINDED_PARTIAL` if the blinding public key and all or only some blinding fields respectively are present.

Note: Returns `WALLY_ERROR` if the value or asset tag blinding key is invalid.

Returns See *Error Codes*

```
int wally_psbt_init_alloc (uint32_t version, size_t inputs_allocation_len, size_t out-
                          puts_allocation_len, size_t global_unknowns_allocation_len,
                          uint32_t flags, struct wally_psbt **output)
```

Allocate and initialize a new PSBT.

Parameters

- **version** – The version of the PSBT. Must be WALLY_PSBT_VERSION_0 or WALLY_PSBT_VERSION_2.
- **inputs_allocation_len** – The number of inputs to pre-allocate space for.
- **outputs_allocation_len** – The number of outputs to pre-allocate space for.
- **global_unknowns_allocation_len** – The number of global unknowns to allocate space for.
- **flags** – Flags controlling psbt creation. Must be 0 or WALLY_PSBT_INIT_PSET.
- **output** – Destination for the resulting PSBT output.

Returns See *Error Codes*

```
int wally_psbt_free (struct wally_psbt *psbt)
    Free a PSBT allocated by wally_psbt_init_alloc.
```

Parameters

- **psbt** – The PSBT to free.

Returns See *Error Codes*

```
int wally_psbt_set_version (struct wally_psbt *psbt, uint32_t flags, uint32_t version)
    Set the version for a PSBT.
```

Parameters

- **psbt** – The PSBT to set the version for.
- **flags** – Flags controlling the version upgrade/downgrade. Must be 0.
- **version** – The version to use for the PSBT. Must be WALLY_PSBT_VERSION_0 or WALLY_PSBT_VERSION_2.

Note: This call converts the PSBT in place to the specified version.

Returns See *Error Codes*

```
int wally_psbt_get_id (const struct wally_psbt *psbt, uint32_t flags, unsigned char *bytes_out,
                      size_t len)
    Return the BIP-370 unique id of a PSBT.
```

Parameters

- **psbt** – The PSBT to compute the id of.
- **flags** – **WALLY_PSBT_ID_** flags to change the id calculation, or pass 0 to compute a BIP-370 compatible id.
- **bytes_out** – Destination for the id.
- **len** – Size of bytes_out. Must be WALLY_TXHASH_LEN.

Note: The id is expensive to compute.

Returns See [Error Codes](#)

int **wally_psbt_get_locktime** (const struct wally_psbt *psbt, size_t *written)
Return the calculated transaction lock time of a PSBT.

Parameters

- **psbt** – The PSBT to compute the lock time of. Must be a v2 PSBT.
- **written** – Destination for the calculated transaction lock time.

Note: The calculated lock time may change as the PSBT is modified.

Returns See [Error Codes](#)

int **wally_psbt_is_finalized** (const struct wally_psbt *psbt, size_t *written)
Determine if all PSBT inputs are finalized.

Parameters

- **psbt** – The PSBT to check.
- **written** – On success, set to one if the PSBT is finalized, otherwise zero.

Returns See [Error Codes](#)

int **wally_psbt_set_global_tx** (struct wally_psbt *psbt, const struct wally_tx *tx)
Set the global transaction for a PSBT.

Parameters

- **psbt** – The PSBT to set the transaction for.
- **tx** – The transaction to set.

The global transaction can only be set on a newly created version 0 PSBT. After this call completes the PSBT will have empty inputs and outputs for each input and output in the transaction tx given.

Returns See [Error Codes](#)

int **wally_psbt_set_tx_version** (struct wally_psbt *psbt, uint32_t version)
Set the transaction version for a PSBT.

Parameters

- **psbt** – The PSBT to set the transaction version for. Must be a v2 PSBT.
- **version** – The version to use for the transaction. Must be at least 2.

Returns See [Error Codes](#)

int **wally_psbt_get_tx_version** (const struct wally_psbt *psbt, size_t *written)
Get the transaction version of a PSBT.

Parameters

- **psbt** – The PSBT to get the transaction version for. Must be v2 PSBT.
- **written** – Destination for the PSBT's transaction version.

Note: Returns the default version 2 if none has been explicitly set.

Returns See [Error Codes](#)

int **wally_psbt_set_fallback_locktime** (struct wally_psbt *psbt, uint32_t locktime)
Set the fallback locktime for a PSBT.

Parameters

- **psbt** – The PSBT to set the fallback locktime for.
- **locktime** – The fallback locktime to set.

Sets the fallback locktime field in the transaction. Cannot be set on V0 PSBTs.

Returns See [Error Codes](#)

int **wally_psbt_clear_fallback_locktime** (struct wally_psbt *psbt)
Clear the fallback locktime for a PSBT.

Parameters

- **psbt** – The PSBT to update.

Returns See [Error Codes](#)

int **wally_psbt_set_tx_modifiable_flags** (struct wally_psbt *psbt, uint32_t flags)
Set the transaction modifiable flags for a PSBT.

Parameters

- **psbt** – The PSBT to set the flags for.
- **flags** – **WALLY_PSBT_TXMOD_** flags indicating what can be modified.

Returns See [Error Codes](#)

int **wally_psbt_set_global_scalars** (struct wally_psbt *psbt, const struct wally_map *map_in)
Set the scalar offsets in a PSBT.

Parameters

- **psbt** – The psbt to update. Must be a PSET.
- **map_in** – The scalar offsets for this PSBT.

Returns See [Error Codes](#)

int **wally_psbt_add_global_scalar** (struct wally_psbt *psbt, const unsigned char *scalar,
size_t scalar_len)
Add a scalar offset to a PSBT.

Parameters

- **psbt** – The PSBT to add to. Must be a PSET.
- **scalar** – The scalar offset to add.
- **scalar_len** – The length of the scalar offset. Must be 32.

Returns See [Error Codes](#)

int **wally_psbt_find_global_scalar** (struct wally_psbt *psbt, const unsigned char *scalar,
size_t scalar_len, size_t *written)
Find a scalar offset in a PSBT.

Parameters

- **psbt** – The PSBT to find in. Must be a PSET.
- **scalar** – The scalar offset to find.
- **scalar_len** – The length of the scalar offset. Must be 32.
- **written** – On success, set to zero if the item is not found, otherwise the index of the item plus one.

Returns See *Error Codes*

int **wally_psbt_set_pset_modifiable_flags** (struct wally_psbt *psbt, uint32_t flags)
Set the Elements transaction modifiable flags for a PSBT.

Parameters

- **psbt** – The PSBT to set the flags for.
- **flags** – **PSBT_ELEMENTS_TX_MODIFIABLE_FLAGS** flags indicating what can be modified.

Returns See *Error Codes*

int **wally_psbt_add_tx_input_at** (struct wally_psbt *psbt, uint32_t index, uint32_t flags, const struct wally_tx_input *input)
Add a transaction input to a PSBT at a given position.

Parameters

- **psbt** – The PSBT to add the input to.
- **index** – The zero-based index of the position to add the input at.
- **flags** – Flags controlling input insertion. Must be 0 or WALLY_PSBT_FLAG_NON_FINAL.
- **input** – The transaction input to add.

Returns See *Error Codes*

int **wally_psbt_remove_input** (struct wally_psbt *psbt, uint32_t index)
Remove a transaction input from a PSBT.

Parameters

- **psbt** – The PSBT to remove the input from.
- **index** – The zero-based index of the input to remove.

Returns See *Error Codes*

int **wally_psbt_get_input_bip32_key_from_alloc** (const struct wally_psbt *psbt, size_t index, size_t subindex, uint32_t flags, const struct ext_key *hdkey, struct ext_key **output)
Return a BIP32 derived key matching a keypath in a PSBT input.

Parameters

- **psbt** – The PSBT containing the input whose keypaths to search.
- **index** – The zero-based index of the input in the PSBT.
- **subindex** – The zero-based index of the keypath to start searching from.
- **flags** – Flags controlling the keypath search. Must be 0.
- **hdkey** – The BIP32 parent key to derive matches from.

- **output** – Destination for the resulting derived key, if any.

Note: See `wally_map_keypath_get_bip32_key_from_alloc`.

Returns See *Error Codes*

int **wally_psbt_get_input_signing_script_len** (const struct wally_psbt *psbt, size_t index, size_t *written)
Get the length of the scriptPubKey or redeem script from a PSBT input.

Parameters

- **psbt** – The PSBT containing the input to get from.
- **index** – The zero-based index of the input to get the script length from.
- **written** – Destination for the length of the script.

Returns See *Error Codes*

int **wally_psbt_get_input_signing_script** (const struct wally_psbt *psbt, size_t index, unsigned char *bytes_out, size_t len, size_t *written)
Get the scriptPubKey or redeem script from a PSBT input.

Parameters

- **psbt** – The PSBT containing the input to get from.
- **index** – The zero-based index of the input to get the script from.
- **bytes_out** – Destination for the scriptPubKey or redeem script.
- **len** – Length of bytes in bytes.
- **written** – Destination for the number of bytes written to bytes_out.

Returns See *Variable Length Output Buffers*

int **wally_psbt_get_input_scriptcode_len** (const struct wally_psbt *psbt, size_t index, const unsigned char *script, size_t script_len, size_t *written)
Get the length of the scriptCode for signing a PSBT input.

Parameters

- **psbt** – The PSBT containing the input to get from.
- **index** – The zero-based index of the input to get the script from.
- **script** – scriptPubKey/redeem script from `wally_psbt_get_input_signing_script`.
- **script_len** – Length of script in bytes.
- **written** – Destination for the length of the scriptCode.

Returns See *Error Codes*

int **wally_psbt_get_input_scriptcode** (const struct wally_psbt *psbt, size_t index, const unsigned char *script, size_t script_len, unsigned char *bytes_out, size_t len, size_t *written)
Get the scriptCode for signing a PSBT input given its scriptPubKey/redeem script.

Parameters

- **psbt** – The PSBT containing the input to get from.
- **index** – The zero-based index of the input to get the script from.

- **script** – scriptPubKey/redeem script from `wally_psbt_get_input_signing_script`.
- **script_len** – Length of `script` in bytes.
- **bytes_out** – Destination for the scriptCode.
- **len** – Length of `bytes` in bytes.
- **written** – Destination for the number of bytes written to `bytes_out`.

Returns See [Variable Length Output Buffers](#)

```
int wally_psbt_get_input_signature_hash (struct wally_psbt *psbt, size_t index, const
                                         struct wally_tx *tx, const unsigned char *script,
                                         size_t script_len, uint32_t flags, unsigned
                                         char *bytes_out, size_t len)
```

Create a transaction for signing a PSBT input and return its hash.

Parameters

- **psbt** – The PSBT containing the input to compute a signature hash for.
- **index** – The zero-based index of the PSBT input to sign.
- **tx** – The transaction to generate the signature hash from.
- **script** – The (unprefixed) scriptCode for the input being signed.
- **script_len** – Length of `script` in bytes.
- **flags** – Flags controlling signature hash generation. Must be 0.
- **bytes_out** – Destination for the signature hash.
- **len** – Size of `bytes_out`. Must be `SHA256_LEN`.

Returns See [Error Codes](#)

```
int wally_psbt_add_tx_output_at (struct wally_psbt *psbt, uint32_t index, uint32_t flags, const struct
                                wally_tx_output *output)
```

Add a transaction output to a PSBT at a given position.

Parameters

- **psbt** – The PSBT to add the output to.
- **index** – The zero-based index of the position to add the output at.
- **flags** – Flags controlling output insertion. Must be 0.
- **output** – The transaction output to add.

Returns See [Error Codes](#)

```
int wally_psbt_remove_output (struct wally_psbt *psbt, uint32_t index)
```

Remove a transaction output from a PSBT.

Parameters

- **psbt** – The PSBT to remove the output from.
- **index** – The zero-based index of the output to remove.

Returns See [Error Codes](#)

```
int wally_psbt_from_bytes (const unsigned char *bytes, size_t bytes_len, uint32_t flags, struct
                           wally_psbt **output)
```

Create a PSBT from its serialized bytes.

Parameters

- **bytes** – Bytes to create the PSBT from.
- **bytes_len** – Length of **bytes** in bytes.
- **flags** – **WALLY_PSBT_PARSE_FLAG** flags controlling deserialization.
- **output** – Destination for the resulting PSBT.

Returns See *Error Codes*

int **wally_psb_get_length** (const struct wally_psb *psbt, uint32_t flags, size_t *written)
Get the length of a PSBT when serialized to bytes.

Parameters

- **psbt** – the PSBT.
- **flags** – Flags controlling length determination. Must be 0.
- **written** – Destination for the length in bytes when serialized.

Returns See *Error Codes*

int **wally_psb_to_bytes** (const struct wally_psb *psbt, uint32_t flags, unsigned char *bytes_out, size_t len, size_t *written)
Serialize a PSBT to bytes.

Parameters

- **psbt** – the PSBT to serialize.
- **flags** – Flags controlling serialization. Must be 0.
- **bytes_out** – Destination for the serialized PSBT.
- **len** – Length of **bytes** in bytes (use *wally_psb_get_length*).
- **written** – number of bytes written to **bytes_out**.

Returns See *Variable Length Output Buffers*

int **wally_psb_from_base64** (const char *base64, uint32_t flags, struct wally_psb **output)
Create a PSBT from its serialized base64 string.

Parameters

- **base64** – Base64 string to create the PSBT from.
- **flags** – **WALLY_PSBT_PARSE_FLAG** flags controlling deserialization.
- **output** – Destination for the resulting PSBT.

Returns See *Error Codes*

int **wally_psb_to_base64** (const struct wally_psb *psbt, uint32_t flags, char **output)
Serialize a PSBT to a base64 string.

Parameters

- **psbt** – the PSBT to serialize.
- **flags** – Flags controlling serialization. Must be 0.
- **output** – Destination for the resulting serialized PSBT.

Returns See *Error Codes*

int **wally_psbt_combine** (struct wally_psbt **psbt*, const struct wally_psbt **source*)

Combine the metadata from a source PSBT into another PSBT.

Parameters

- **psbt** – the PSBT to combine into.
- **source** – the PSBT to copy data from.

Returns See [Error Codes](#)

int **wally_psbt_clone_alloc** (const struct wally_psbt **psbt*, uint32_t *flags*, struct wally_psbt ***output*)

Clone a PSBT into a newly allocated copy.

Parameters

- **psbt** – the PSBT to clone.
- **flags** – Flags controlling PSBT creation. Must be 0.
- **output** – Destination for the resulting cloned PSBT.

Returns See [Error Codes](#)

int **wally_psbt_blind** (struct wally_psbt **psbt*, const struct wally_map **values*, const struct wally_map **vbfs*, const struct wally_map **assets*, const struct wally_map **abfs*, const unsigned char **entropy*, size_t *entropy_len*, uint32_t *output_index*, uint32_t *flags*, struct wally_map **output*)

Blind a PSBT.

Parameters

- **psbt** – PSBT to blind. Directly modifies this PSBT.
- **values** – Integer map of input index to value for the callers inputs.
- **vbfs** – Integer map of input index to value blinding factor for the callers inputs.
- **assets** – Integer map of input index to asset tags for the callers inputs.
- **abfs** – Integer map of input index to asset blinding factors for the callers inputs.
- **entropy** – Random entropy for asset and blinding factor generation.
- **entropy_len** – Size of *entropy* in bytes. Must be a multiple of 5 * BLINDING_FACTOR_LEN for each non-fee output to be blinded, with an additional 2 * BLINDING_FACTOR_LEN bytes for any issuance outputs.
- **output_index** – The zero based index of the output to blind, or WALLY_PSET_BLIND_ALL.
- **flags** – Flags controlling blinding. Must be 0.
- **output** – Destination for a map of integer output index to the ephemeral private key used to blind the output. Ignored if NULL.

Returns See [Error Codes](#)

int **wally_psbt_blind_alloc** (struct wally_psbt **psbt*, const struct wally_map **values*, const struct wally_map **vbfs*, const struct wally_map **assets*, const struct wally_map **abfs*, const unsigned char **entropy*, size_t *entropy_len*, uint32_t *output_index*, uint32_t *flags*, struct wally_map ***output*)

Blind a PSBT.

As per [wally_psbt_blind](#), but allocates the output map.

Returns See [Error Codes](#)

int **wally_psbt_sign** (struct wally_psbt *psbt, const unsigned char *key, size_t key_len, uint32_t flags)
Sign PSBT inputs corresponding to a given private key.

Parameters

- **psbt** – PSBT to sign. Directly modifies this PSBT.
- **key** – Private key to sign PSBT with.
- **key_len** – Length of key in bytes. Must be EC_PRIVATE_KEY_LEN.
- **flags** – Flags controlling signing. Must be 0 or EC_FLAG_GRIND_R.

Note: See <https://github.com/bitcoin/bips/blob/master/bip-0174.mediawiki#simple-signer-algorithm> for a description of the signing algorithm.

Returns See *Error Codes*

int **wally_psbt_sign_bip32** (struct wally_psbt *psbt, const struct ext_key *hdkey, uint32_t flags)
Sign PSBT inputs corresponding to a given BIP32 parent key.

Parameters

- **psbt** – PSBT to sign. Directly modifies this PSBT.
- **hdkey** – The parent extended key to derive signing keys from.
- **flags** – Flags controlling signing. Must be 0 or EC_FLAG_GRIND_R.

Note: See <https://github.com/bitcoin/bips/blob/master/bip-0174.mediawiki#simple-signer-algorithm> for a description of the signing algorithm.

Returns See *Error Codes*

int **wally_psbt_sign_input_bip32** (struct wally_psbt *psbt, size_t index, size_t subindex, const unsigned char *txhash, size_t txhash_len, const struct ext_key *hdkey, uint32_t flags)
Sign a single PSBT input with a given BIP32 key.

Parameters

- **psbt** – PSBT containing the input to sign. Directly modifies this PSBT.
- **index** – The zero-based index of the input in the PSBT.
- **subindex** – The zero-based index of the keypath to start searching from.
- **txhash** – The signature hash to sign, from *wally_psbt_get_input_signature_hash*.
- **txhash_len** – Size of txhash in bytes. Must be WALLY_TXHASH_LEN.
- **hdkey** – The derived extended key to sign with.
- **flags** – Flags controlling signing. Must be 0 or EC_FLAG_GRIND_R.

Returns See *Error Codes*

int **wally_psbt_finalize** (struct wally_psbt *psbt)
Finalize a PSBT.

Parameters

- **psbt** – PSBT to finalize. Directly modifies this PSBT.

Returns See [Error Codes](#)

int **wally_psbt_extract** (const struct wally_psbt **psbt*, uint32_t *flags*, struct wally_tx ***output*)

Extract a network transaction from a finalized PSBT.

Parameters

- **psbt** – PSBT to extract from.
- **flags** – Flags controlling signing. Must be 0 or WALLY_PSBT_EXTRACT_NON_FINAL.
- **output** – Destination for the resulting transaction.

Returns See [Error Codes](#)

int **wally_psbt_is_elements** (const struct wally_psbt **psbt*, size_t **written*)

Determine if a PSBT is an elements PSBT.

Parameters

- **psbt** – The PSBT to check.
- **written** – 1 if the PSBT is an elements PSBT, otherwise 0.

Returns See [Error Codes](#)

Script Functions

int **wally_scriptpubkey_get_type** (const unsigned char *bytes, size_t bytes_len, size_t *written)
Determine the type of a scriptPubkey script.

Parameters

- **bytes** – Bytes of the scriptPubkey.
- **bytes_len** – Length of bytes in bytes.
- **written** – Destination for the WALLY_SCRIPT_TYPE_ script type.

Returns See *Error Codes*

int **wally_scriptpubkey_p2pkh_from_bytes** (const unsigned char *bytes, size_t bytes_len,
uint32_t flags, unsigned char *bytes_out, size_t len,
size_t *written)

Create a P2PKH scriptPubkey.

Parameters

- **bytes** – Bytes to create a scriptPubkey for.
- **bytes_len** – The length of bytes in bytes. If WALLY_SCRIPT_HASH160 is given in flags, bytes is a public key to hash160 before creating the P2PKH, and bytes_len must be EC_PUBLIC_KEY_LEN or EC_PUBLIC_KEY_UNCOMPRESSED_LEN. Otherwise, bytes_len must be HASH160_LEN and bytes must contain the hash160 to use.
- **flags** – WALLY_SCRIPT_HASH160 or 0.
- **bytes_out** – Destination for the resulting scriptPubkey.
- **n** – Size of bytes_out. Passing WALLY_SCRIPTPUBKEY_P2PKH_LEN will ensure the buffer is large enough.
- **written** – Destination for the number of bytes written to bytes_out.

Returns See *Variable Length Output Buffers*

```
int wally_scriptsig_p2pkh_from_sig (const unsigned char *pub_key, size_t pub_key_len, const unsigned char *sig, size_t sig_len, uint32_t sighash, unsigned char *bytes_out, size_t len, size_t *written)
```

Create a P2PKH scriptSig from a pubkey and compact signature.

This function creates the scriptSig by converting `sig` to DER encoding, appending the given sighash, then calling `wally_scriptsig_p2pkh_from_der`.

Parameters

- **pub_key** – The public key to create a scriptSig with.
- **pub_key_len** – Length of `pub_key` in bytes. Must be `EC_PUBLIC_KEY_LEN` or `EC_PUBLIC_KEY_UNCOMPRESSED_LEN`.
- **sig** – The compact signature to create a scriptSig with.
- **sig_len** – The length of `sig` in bytes. Must be `EC_SIGNATURE_LEN`.
- **sighash** – `WALLY_SIGHASH_` flags specifying the type of signature desired.
- **bytes_out** – Destination for the resulting scriptSig.
- **len** – The length of `bytes_out` in bytes.
- **n** – Size of `bytes_out`. Passing `WALLY_SCRIPTSIG_P2PKH_MAX_LEN` will ensure the buffer is large enough.
- **written** – Destination for the number of bytes written to `bytes_out`.

Returns See *Variable Length Output Buffers*

```
int wally_witness_p2wpkh_from_sig (const unsigned char *pub_key, size_t pub_key_len, const unsigned char *sig, size_t sig_len, uint32_t sighash, struct wally_tx_witness_stack **witness)
```

Create a P2WPKH witness from a pubkey and compact signature.

Parameters

- **pub_key** – The public key to create a witness with.
- **pub_key_len** – Length of `pub_key` in bytes. Must be `EC_PUBLIC_KEY_LEN` or `EC_PUBLIC_KEY_UNCOMPRESSED_LEN`.
- **sig** – The compact signature to create a witness with.
- **sig_len** – The length of `sig` in bytes. Must be `EC_SIGNATURE_LEN`.
- **sighash** – `WALLY_SIGHASH_` flags specifying the type of signature desired.
- **witness** – Destination for the newly created witness.

Returns See *Error Codes*

```
int wally_scriptsig_p2pkh_from_der (const unsigned char *pub_key, size_t pub_key_len, const unsigned char *sig, size_t sig_len, unsigned char *bytes_out, size_t len, size_t *written)
```

Create a P2PKH scriptSig from a pubkey and DER signature plus sighash.

Parameters

- **pub_key** – The public key to create a scriptSig with.
- **pub_key_len** – Length of `pub_key` in bytes. Must be `EC_PUBLIC_KEY_LEN` or `EC_PUBLIC_KEY_UNCOMPRESSED_LEN`.

- **sig** – The DER encoded signature to create a scriptSig, with the sighash byte appended to it.
- **sig_len** – The length of **sig** in bytes.
- **bytes_out** – Destination for the resulting scriptSig.
- **n** – Size of **bytes_out**. Passing `WALLY_SCRIPTSIG_P2PKH_MAX_LEN` will ensure the buffer is large enough.
- **written** – Destination for the number of bytes written to **bytes_out**.

Returns See *Variable Length Output Buffers*

```
int wally_witness_p2wpkh_from_der (const unsigned char *pub_key, size_t pub_key_len,
                                   const unsigned char *sig, size_t sig_len, struct
                                   wally_tx_witness_stack **witness)
Create a P2WPKH witness from a pubkey and DER signature plus sighash.
```

Parameters

- **pub_key** – The public key to create a witness with.
- **pub_key_len** – Length of **pub_key** in bytes. Must be `EC_PUBLIC_KEY_LEN` or `EC_PUBLIC_KEY_UNCOMPRESSED_LEN`.
- **sig** – The DER encoded signature to create a witness, with the sighash byte appended to it.
- **sig_len** – The length of **sig** in bytes.
- **witness** – Destination for the newly created witness.

Returns See *Error Codes*

```
int wally_scriptpubkey_op_return_from_bytes (const unsigned char *bytes, size_t bytes_len,
                                             uint32_t flags, unsigned char *bytes_out,
                                             size_t len, size_t *written)
Create an OP_RETURN scriptPubkey.
```

Parameters

- **bytes** – Bytes to create a scriptPubkey for.
- **bytes_len** – Length of **bytes** in bytes. Must be less than or equal to `WALLY_MAX_OP_RETURN_LEN`.
- **flags** – Currently unused, must be 0.
- **bytes_out** – Destination for the resulting scriptPubkey.
- **n** – Size of **bytes_out**. Passing `WALLY_SCRIPTPUBKEY_OP_RETURN_MAX_LEN` will ensure the buffer is large enough.
- **written** – Destination for the number of bytes written to **bytes_out**.

Returns See *Variable Length Output Buffers*

```
int wally_scriptpubkey_p2sh_from_bytes (const unsigned char *bytes, size_t bytes_len,
                                         uint32_t flags, unsigned char *bytes_out, size_t len,
                                         size_t *written)
Create a P2SH scriptPubkey.
```

Parameters

- **bytes** – Bytes to create a scriptPubkey for. If `WALLY_SCRIPT_HASH160` is given, **bytes** is a redeem script to hash160 before creating the P2SH. Otherwise, **bytes_len** must be `HASH160_LEN` and **bytes** must contain the hash160 to use.

- **bytes_len** – Length of `bytes` in bytes.
- **flags** – `WALLY_SCRIPT_HASH160` or `0`.
- **bytes_out** – Destination for the resulting scriptPubkey.
- **n** – Size of `bytes_out`. Passing `WALLY_SCRIPTPUBKEY_P2SH_LEN` will ensure the buffer is large enough.
- **written** – Destination for the number of bytes written to `bytes_out`.

Returns See *Variable Length Output Buffers*

```
int wally_scriptpubkey_multisig_from_bytes (const unsigned char *bytes, size_t bytes_len,
                                           uint32_t threshold, uint32_t flags, unsigned
                                           char *bytes_out, size_t len, size_t *written)
```

Create a multisig scriptPubkey.

Parameters

- **bytes** – Compressed public keys to create a scriptPubkey from.
- **bytes_len** – Length of `bytes` in bytes. Must be a multiple of `EC_PUBLIC_KEY_LEN`.
- **threshold** – The number of signatures that must match to satisfy the script.
- **flags** – Must be `WALLY_SCRIPT_MULTISIG_SORTED` for BIP67 sorting or `0`.
- **bytes_out** – Destination for the resulting scriptPubkey.
- **len** – The length of `bytes_out` in bytes.
- **written** – Destination for the number of bytes written to `bytes_out`.

Note: A maximum of 15 keys are allowed to be passed.

Returns See *Variable Length Output Buffers*

```
int wally_scriptsig_multisig_from_bytes (const unsigned char *script, size_t script_len,
                                         const unsigned char *bytes, size_t bytes_len, const
                                         uint32_t *sighash, size_t sighash_len, uint32_t flags,
                                         unsigned char *bytes_out, size_t len, size_t *written)
```

Create a multisig scriptSig.

Parameters

- **script** – The redeem script this scriptSig provides signatures for.
- **script_len** – The length of `script` in bytes.
- **bytes** – Compact signatures to place in the scriptSig.
- **bytes_len** – Length of `bytes` in bytes. Must be a multiple of `EC_SIGNATURE_LEN`.
- **sighash** – `WALLY_SIGHASH_` flags for each signature in `bytes`.
- **sighash_len** – The number of sighash flags in `sighash`.
- **flags** – Must be zero.
- **bytes_out** – Destination for the resulting scriptSig.
- **len** – The length of `bytes_out` in bytes.
- **written** – Destination for the number of bytes written to `bytes_out`.

Returns See *Variable Length Output Buffers*

```
int wally_witness_multisig_from_bytes (const unsigned char *script, size_t script_len,
                                     const unsigned char *bytes, size_t bytes_len, const
                                     uint32_t *sighash, size_t sighash_len, uint32_t flags,
                                     struct wally_tx_witness_stack **witness)
```

Create a multisig scriptWitness.

Parameters

- **script** – The witness script this scriptWitness provides signatures for.
- **script_len** – The length of script in bytes.
- **bytes** – Compact signatures to place in the scriptWitness.
- **bytes_len** – Length of bytes in bytes. Must be a multiple of EC_SIGNATURE_LEN.
- **sighash** – WALLY_SIGHASH_ flags for each signature in bytes.
- **sighash_len** – The number of sighash flags in sighash.
- **flags** – Must be zero.
- **witness** – Destination for newly allocated witness.

Returns See *Error Codes*

```
int wally_scriptpubkey_csv_2of2_then_1_from_bytes (const unsigned char *bytes,
                                                  size_t bytes_len, uint32_t csv_blocks,
                                                  uint32_t flags, unsigned
                                                  char *bytes_out, size_t len,
                                                  size_t *written)
```

Create a CSV 2of2 multisig with a single key recovery scriptPubkey.

The resulting output can be spent at any time with both of the two keys given, and by the last (recovery) key alone, csv_blocks after the output confirms.

Parameters

- **bytes** – Compressed public keys to create a scriptPubkey from. The second key given will be used as the recovery key.
- **bytes_len** – Length of bytes in bytes. Must 2 * EC_PUBLIC_KEY_LEN.
- **csv_blocks** – The number of blocks before the recovery key can be used. Must be between 17 and 65536.
- **flags** – Must be zero.
- **bytes_out** – Destination for the resulting scriptPubkey.
- **len** – The length of bytes_out in bytes.
- **written** – Destination for the number of bytes written to bytes_out.

Returns See *Variable Length Output Buffers*

```
int wally_scriptpubkey_csv_2of2_then_1_from_bytes_opt (const unsigned char *bytes,
                                                       size_t bytes_len,
                                                       uint32_t csv_blocks,
                                                       uint32_t flags, unsigned
                                                       char *bytes_out, size_t len,
                                                       size_t *written)
```

Create an optimised CSV 2of2 multisig with a single key recovery scriptPubkey.

Works like `wally_scriptpubkey_csv_2of2_then_1_from_bytes` but produces a script that is smaller and compatible with the miniscript expression “`and(pk(key_user),or(99@pk(key_service),older(<csv_blocks>)))`”.

Returns See *Variable Length Output Buffers*

```
int wally_scriptpubkey_csv_2of3_then_2_from_bytes (const unsigned char *bytes,
                                                    size_t bytes_len, uint32_t csv_blocks,
                                                    uint32_t flags, unsigned
                                                    char *bytes_out, size_t len,
                                                    size_t *written)
```

Create a CSV 2of3 multisig with two key recovery scriptPubkey.

The resulting output can be spent at any time with any two of the three keys given, and by either of the last two (recovery) keys alone, `csv_blocks` after the output confirms.

Parameters

- **bytes** – Compressed public keys to create a scriptPubkey from. The second and third keys given will be used as the recovery keys.
- **bytes_len** – Length of `bytes` in bytes. Must $3 * EC_PUBLIC_KEY_LEN$.
- **csv_blocks** – The number of blocks before the recovery keys can be used. Must be between 17 and 65536.
- **flags** – Must be zero.
- **bytes_out** – Destination for the resulting scriptPubkey.
- **len** – The length of `bytes_out` in bytes.
- **written** – Destination for the number of bytes written to `bytes_out`.

Returns See *Variable Length Output Buffers*

```
int wally_script_push_from_bytes (const unsigned char *bytes, size_t bytes_len, uint32_t flags, un-
                                   signed char *bytes_out, size_t len, size_t *written)
```

Create a bitcoin script that pushes data to the stack.

Parameters

- **bytes** – Bytes to create a push script for.
- **bytes_len** – Length of `bytes` in bytes.
- **flags** – `WALLY_SCRIPT_HASH160` or `WALLY_SCRIPT_SHA256` to hash `bytes` before pushing it.
- **bytes_out** – Destination for the resulting push script.
- **len** – The length of `bytes_out` in bytes.
- **written** – Destination for the number of bytes written to `bytes_out`.

Returns See *Variable Length Output Buffers*

```
int wally_varint_get_length (uint64_t value, size_t *written)
```

Get the length of an integer serialized to a varint.

Parameters

- **value** – The integer value to be find the length of.
- **written** – Destination for the length of the integer when serialized.

Returns See *Error Codes*

int **wally_varint_to_bytes** (uint64_t *value*, unsigned char **bytes_out*, size_t *len*, size_t **written*)
Serialize an integer to a buffer as a varint.

Parameters

- **value** – The integer value to be serialized.
- **bytes_out** – Destination for the resulting serialized varint.
- **len** – The length of *bytes_out* in bytes. Must be at least `wally_varint_get_length(value)`.
- **written** – Destination for the number of bytes written to *bytes_out*.

Returns See [Variable Length Output Buffers](#)

int **wally_varbuff_get_length** (const unsigned char **bytes*, size_t *bytes_len*, size_t **written*)
Get the length of a buffer serialized to a varbuff (varint size, followed by the buffer).

Parameters

- **bytes** – The buffer to get the length of.
- **bytes_len** – Length of *bytes* in bytes.
- **written** – Destination for the length of the buffer when serialized.

Returns See [Error Codes](#)

int **wally_varbuff_to_bytes** (const unsigned char **bytes*, size_t *bytes_len*, unsigned char **bytes_out*,
size_t *len*, size_t **written*)
Serialize a buffer to a varbuff (varint size, followed by the buffer).

Parameters

- **bytes** – The buffer to be serialized.
- **bytes_len** – Length of *bytes* in bytes.
- **bytes_out** – Destination for the resulting serialized varbuff.
- **len** – The length of *bytes_out* in bytes. Must be at least `wally_varbuff_get_length(bytes, bytes_len)`.
- **written** – Destination for the number of bytes written to *bytes_out*.

Returns See [Variable Length Output Buffers](#)

int **wally_witness_program_from_bytes** (const unsigned char **bytes*, size_t *bytes_len*,
uint32_t *flags*, unsigned char **bytes_out*, size_t *len*,
size_t **written*)
Create a segwit witness program from a script or hash.

Parameters

- **bytes** – Script or hash bytes to create a witness program from.
- **bytes_len** – Length of *bytes* in bytes. Must be `HASH160_LEN` or `SHA256_LEN` if neither `WALLY_SCRIPT_HASH160` or `WALLY_SCRIPT_SHA256` is given.
- **flags** – `WALLY_SCRIPT_HASH160` or `WALLY_SCRIPT_SHA256` to hash the input script before using it. `WALLY_SCRIPT_AS_PUSH` to generate a push of the generated script as used for the scriptSig in p2sh-p2wpkh and p2sh-p2wsh.
- **bytes_out** – Destination for the resulting witness program.
- **n** – Size of *bytes_out*. Passing `WALLY_WITNESSSCRIPT_MAX_LEN` will ensure the buffer is large enough.

- **written** – Destination for the number of bytes written to `bytes_out`.

Returns See *Variable Length Output Buffers*

```
int wally_witness_program_from_bytes_and_version (const unsigned char *bytes,
                                                  size_t bytes_len, uint32_t version,
                                                  uint32_t flags, unsigned char *bytes_out,
                                                  size_t len, size_t *written)
```

Create a segwit witness program from a script or hash using witness version.

Parameters

- **bytes** – Script or hash bytes to create a witness program from.
- **bytes_len** – Length of `bytes` in bytes.
- **version** – Witness version to create a witness program from. Specify a value of 16 or less.
- **flags** – `WALLY_SCRIPT_HASH160` or `WALLY_SCRIPT_SHA256` to hash the input script before using it. `WALLY_SCRIPT_AS_PUSH` to generate a push of the generated script as used for the scriptSig in p2sh-p2wpkh and p2sh-p2wsh.
- **bytes_out** – Destination for the resulting witness program.
- **n** – Size of `bytes_out`. Passing `WALLY_WITNESSSCRIPT_MAX_LEN` will ensure the buffer is large enough.
- **written** – Destination for the number of bytes written to `bytes_out`.

Returns See *Variable Length Output Buffers*

```
int wally_elements_pegout_script_size (size_t genesis_blockhash_len, size_t mainchain_script_len,
                                       size_t sub_pubkey_len,
                                       size_t whitelistproof_len, size_t *written)
```

Get the pegout script size.

Parameters

- **genesis_blockhash_len** – Length of `genesis_blockhash` in bytes. Must be `SHA256_LEN`.
- **mainchain_script_len** – Length of `mainchain_script` in bytes.
- **sub_pubkey_len** – Length of `sub_pubkey` in bytes. Must be `EC_PUBLIC_KEY_LEN`.
- **whitelistproof_len** – The length of `whitelistproof` in bytes.
- **written** – Destination for the number of bytes required to hold the pegout script.

Returns See *Error Codes*

```
int wally_elements_pegout_script_from_bytes (const unsigned char *genesis_blockhash,
                                              size_t genesis_blockhash_len, const unsigned char *mainchain_script,
                                              size_t mainchain_script_len, const unsigned char *sub_pubkey,
                                              size_t sub_pubkey_len, const unsigned char *whitelistproof,
                                              size_t whitelistproof_len, uint32_t flags, unsigned char *bytes_out,
                                              size_t len, size_t *written)
```

Create a pegout script.

Parameters

- **genesis_blockhash** – The genesis blockhash of the parent chain.
- **genesis_blockhash_len** – Length of `genesis_blockhash` in bytes. Must be `SHA256_LEN`.
- **mainchain_script** – The parent chain script.
- **mainchain_script_len** – Length of `mainchain_script` in bytes.
- **sub_pubkey** – The whitelisted public key.
- **sub_pubkey_len** – Length of `sub_pubkey` in bytes. Must be `EC_PUBLIC_KEY_LEN`.
- **whitelistproof** – The whitelist proof.
- **whitelistproof_len** – The length of `whitelistproof` in bytes.
- **flags** – Must be zero.
- **bytes_out** – Destination for the resulting pegout script.
- **len** – The length of `bytes_out` in bytes.
- **written** – Destination for the number of bytes written to `bytes_out`.

Returns See *Variable Length Output Buffers*

```
int wally_elements_pegin_contract_script_from_bytes (const unsigned char *redeem_script, size_t redeem_script_len, const unsigned char *script, size_t script_len, uint32_t flags, unsigned char *bytes_out, size_t len, size_t *written)
```

Create a script for P2CH peg-in transactions.

Parameters

- **redeem_script** – The federation redeem script.
- **redeem_script_len** – Length of `redeem_script` in bytes.
- **script** – The claim script.
- **script_len** – Length of `script` in bytes.
- **flags** – Must be zero.
- **bytes_out** – Destination for the resulting script.
- **len** – Length of `bytes_out` in bytes.
- **written** – Destination for the number of bytes written to `bytes_out`.

Returns See *Variable Length Output Buffers*

Descriptor Functions

int **wally_descriptor_parse** (const char **descriptor*, const struct wally_map **vars_in*, uint32_t *network*, uint32_t *flags*, struct wally_descriptor ***output*)
Parse an output descriptor or miniscript expression.

Parameters

- **descriptor** – Output descriptor or miniscript expression to parse.
- **vars_in** – Map of variable names to values, or NULL.
- **network** – WALLY_NETWORK_ constant describing the network the descriptor belongs to, or WALLY_NETWORK_NONE for miniscript-only expressions.
- **flags** – Include WALLY_MINISCRIP_ONLY to disallow descriptor expressions, WALLY_MINISCRIP_TAPSCRIPT to use x-only pubkeys, or 0.
- **output** – Destination for the resulting parsed descriptor. The descriptor returned should be freed using *wally_descriptor_free*.

Variable names can be used in the descriptor string and will be substituted with the contents of *vars_in* during parsing. Key names for *vars_in* must be 16 characters or less in length and start with a letter.

Returns See *Error Codes*

int **wally_descriptor_free** (struct wally_descriptor **descriptor*)
Free a parsed output descriptor or miniscript expression.

Parameters

- **descriptor** – Parsed output descriptor or miniscript expression to free.

Returns See *Error Codes*

int **wally_descriptor_canonicalize** (const struct wally_descriptor **descriptor*, uint32_t *flags*, char ***output*)
Canonicalize a descriptor.

Parameters

- **descriptor** – Parsed output descriptor or miniscript expression.

- **flags** – For future use. Must be 0.
- **output** – Destination for the resulting canonical descriptor. The string returned should be freed using *wally_free_string*.

Note: Other canonicalization (hardened derivation indicator)

mapping, and private to public key mapping) is not yet implemented.

Returns See *Error Codes*

int **wally_descriptor_get_checksum**(const struct wally_descriptor **descriptor*, uint32_t *flags*, char ***output*)

Create an output descriptor checksum.

Parameters

- **descriptor** – Parsed output descriptor or miniscript expression.
- **flags** – For future use. Must be 0.
- **output** – Destination for the resulting descriptor checksum. The string returned should be freed using *wally_free_string*.

Returns See *Error Codes*

int **wally_descriptor_get_network**(const struct wally_descriptor **descriptor*, uint32_t **value_out*)

Get the network used in a parsed output descriptor or miniscript expression.

Parameters

- **descriptor** – Parsed output descriptor or miniscript expression.
- **value_out** – Destination for the resulting WALLY_NETWORK network.

A descriptor parsed with WALLY_NETWORK_NONE will infer its network from the contained key expressions. If the descriptor does not contain network information (e.g. its keys are raw keys), then this function will return WALLY_NETWORK_NONE, and *wally_descriptor_set_network* must be called to set a network for the descriptor before addresses can be generated from it.

Returns See *Error Codes*

int **wally_descriptor_set_network**(struct wally_descriptor **descriptor*, uint32_t *network*)

set the network for a parsed output descriptor or miniscript expression.

Parameters

- **descriptor** – Parsed output descriptor or miniscript expression.
- **network** – The WALLY_NETWORK network constant describing the network the descriptor should belong to.

Note: The network can only be set if it is currently WALLY_NETWORK_NONE.

Returns See *Error Codes*

int **wally_descriptor_get_features**(const struct wally_descriptor **descriptor*, uint32_t **value_out*)

Get the features used in a parsed output descriptor or miniscript expression.

Parameters

- **descriptor** – Parsed output descriptor or miniscript expression.
- **value_out** – Destination for the resulting `WALLY_MS_` feature flags.

Returns See *Error Codes*

```
int wally_descriptor_get_num_variants (const struct wally_descriptor *descriptor,  
                                       uint32_t *value_out)
```

Get the number of variants in a parsed output descriptor or miniscript expression.

Parameters

- **descriptor** – Parsed output descriptor or miniscript expression.
- **value_out** – Destination for the number of available variants.

Expressions such as `combo()` can return more than one script/address for a given key or key path. Each available type is represented as a variant numbered from zero. The variant is passed to `wally_descriptor_to_script` or `wally_descriptor_to_addresses` to generate a script/address corresponding to that variant type.

For `combo()`, the variants are `p2pk`, `p2pkh`, `p2wpkh`, and `p2sh-p2wpkh` in order from 0-3. If the expression's key is uncompressed, only the first two variants are available.

Returns See *Error Codes*

```
int wally_descriptor_get_num_paths (const struct wally_descriptor *descriptor,  
                                    uint32_t *value_out)
```

Get the number of multi-path items in a descriptors path expression(s).

Parameters

- **descriptor** – Parsed output descriptor or miniscript expression.
- **value_out** – Destination for the number of ranges.

Paths in descriptor key expressions can contain multiple paths in the format `<x;y;z>`, where each item corresponds to a new path with the given path elements being one of `x`, `y`, or `z` respectively. The index of the multi-path is passed to `wally_descriptor_to_script` or `wally_descriptor_to_addresses` to generate a script/address corresponding to the corresponding key path.

Note: This function is a stub, multi-path keys are not yet implemented and will currently fail to parse.

Returns See *Error Codes*

```
int wally_descriptor_to_script_get_maximum_length (const struct wally_descriptor *descriptor,  
                                                  uint32_t flags, size_t *written)
```

Get the maximum length of a script corresponding to an output descriptor.

Parameters

- **descriptor** – Parsed output descriptor or miniscript expression.
- **flags** – For future use. Must be 0.
- **written** – Destination for the resulting maximum script length.

Note: This function overestimates the script size required, but is cheap to compute (does not require script generation).

Returns See *Error Codes*

```
int wally_descriptor_to_script (const struct wally_descriptor *descriptor, uint32_t depth,
                               uint32_t index, uint32_t variant, uint32_t multi_index,
                               uint32_t child_num, uint32_t flags, unsigned char *bytes_out,
                               size_t len, size_t *written)
```

Create a script corresponding to an output descriptor or miniscript expression.

Parameters

- **descriptor** – Parsed output descriptor or miniscript expression.
- **depth** – Depth of the expression tree to generate from. Pass 0 to generate from the root.
- **index** – The zero-based index of the child at depth `depth` to generate from.
- **variant** – The variant of descriptor to generate. See [wally_descriptor_get_num_variants](#).
- **multi_index** – The multi-path item to generate. See [wally_descriptor_get_num_paths](#).
- **child_num** – The BIP32 child number to derive, or 0 for static descriptors.
- **flags** – For future use. Must be 0.
- **bytes_out** – Destination for the resulting scriptPubKey or script.
- **len** – The length of `bytes_out` in bytes.
- **written** – Destination for the number of bytes written to `bytes_out`.

Note: For miniscript expressions, the script generated is untyped bitcoin script. For descriptors, a scriptPubKey is generated.

Note: `depth` and `index` can be used to generate sub-scripts from the expression tree. These are expected to be useful once introspection of the expression is improved, e.g. to allow generating the nested script inside `sh()` or `wsh()` expressions.

Returns See [Variable Length Output Buffers](#)

```
int wally_descriptor_to_address (const struct wally_descriptor *descriptor, uint32_t variant,
                                uint32_t multi_index, uint32_t child_num, uint32_t flags,
                                char **output)
```

Create an address corresponding to an output descriptor.

Parameters

- **descriptor** – Parsed output descriptor.
- **variant** – The variant of descriptor to generate. See [wally_descriptor_get_num_variants](#).
- **multi_index** – The multi-path item to generate. See [wally_descriptor_get_num_paths](#).
- **child_num** – The BIP32 child number to derive, or zero for static descriptors.
- **flags** – For future use. Must be 0.
- **output** – Destination for the resulting addresss. The string returned should be freed using [wally_free_string](#).

Returns See *Error Codes*

int **wally_descriptor_to_addresses** (const struct wally_descriptor **descriptor*, uint32_t *variant*,
uint32_t *multi_index*, uint32_t *child_num*, uint32_t *flags*,
char ***output*, size_t *num_outputs*)

Create addresses that correspond to the derived range of an output descriptor.

Parameters

- **descriptor** – Parsed output descriptor.
- **variant** – The variant of descriptor to generate. See *wally_descriptor_get_num_variants*.
- **multi_index** – The multi-path item to generate. See *wally_descriptor_get_num_paths*.
- **child_num** – The BIP32 child number to derive, or zero for static descriptors.
- **flags** – For future use. Must be 0.
- **output** – Destination for the resulting addresses.
- **num_outputs** – The number of items in *output*. Addresses will be generated into this array starting from *child_num*, incrementing by 1. The addresses returned should be freed using *wally_free_string*.

Returns See *Error Codes*

Symmetric Functions

int **wally_symmetric_key_from_seed**(const unsigned char *bytes, size_t bytes_len, unsigned char *bytes_out, size_t len)

Create a new symmetric parent key from entropy.

Parameters

- **bytes** – Entropy to use.
- **bytes_len** – Size of bytes in bytes. Must be one of BIP32_ENTROPY_LEN_128, BIP32_ENTROPY_LEN_256 or BIP32_ENTROPY_LEN_512.
- **bytes_out** – Destination for the resulting parent key.
- **len** – Size of bytes_out. Must be HMAC_SHA512_LEN.

Returns See [Error Codes](#)

int **wally_symmetric_key_from_parent**(const unsigned char *bytes, size_t bytes_len, uint32_t version, const unsigned char *label, size_t label_len, unsigned char *bytes_out, size_t len)

Create a new child symmetric key from a parent key.

Parameters

- **bytes** – Parent key to use.
- **bytes_len** – Size of bytes in bytes. Must be HMAC_SHA512_LEN.
- **version** – Version byte to prepend to label. Must be zero.
- **label** – Label to use for the child.
- **label_len** – Size of label in bytes.
- **bytes_out** – Destination for the resulting key.
- **len** – Size of bytes_out. Must be HMAC_SHA512_LEN.

Returns See [Error Codes](#)

Transaction Functions

int **wally_tx_witness_stack_init_alloc** (size_t *allocation_len*, struct wally_tx_witness_stack ***output*)

Allocate and initialize a new witness stack.

Parameters

- **allocation_len** – The number of items to pre-allocate space for.
- **output** – Destination for the resulting witness stack.

Returns See *Error Codes*

int **wally_tx_witness_stack_clone_alloc** (const struct wally_tx_witness_stack **stack*, struct wally_tx_witness_stack ***output*)

Create a copy of a witness stack.

Parameters

- **stack** – The witness stack to copy.
- **output** – Destination for the resulting copy.

Returns See *Error Codes*

int **wally_tx_witness_stack_add** (struct wally_tx_witness_stack **stack*, const unsigned char **witness*, size_t *witness_len*)

Add a witness to a witness stack.

Parameters

- **stack** – The witness stack to add to.
- **witness** – The witness data to add to the stack.
- **witness_len** – Length of *witness* in bytes.

Returns See *Error Codes*

int **wally_tx_witness_stack_add_dummy** (struct wally_tx_witness_stack **stack*, uint32_t *flags*)

Add a dummy witness item to a witness stack.

Parameters

- **stack** – The witness stack to add to.
- **flags** – `WALLY_TX_DUMMY_` Flags indicating the type of dummy to add.

Returns See [Error Codes](#)

int **wally_tx_witness_stack_set** (struct wally_tx_witness_stack *stack, size_t index, const unsigned char *witness, size_t witness_len)

Set a witness item to a witness stack.

Parameters

- **stack** – The witness stack to add to.
- **index** – Index of the item to set. The stack will grow if needed to this many items.
- **witness** – The witness data to add to the stack.
- **witness_len** – Length of witness in bytes.

Returns See [Error Codes](#)

int **wally_tx_witness_stack_set_dummy** (struct wally_tx_witness_stack *stack, size_t index, uint32_t flags)

Set a dummy witness item to a witness stack.

Parameters

- **stack** – The witness stack to add to.
- **index** – Index of the item to set. The stack will grow if needed to this many items.
- **flags** – `WALLY_TX_DUMMY_` Flags indicating the type of dummy to set.

Returns See [Error Codes](#)

int **wally_tx_witness_stack_free** (struct wally_tx_witness_stack *stack)

Free a transaction witness stack allocated by `wally_tx_witness_stack_init_alloc`.

Parameters

- **stack** – The transaction witness stack to free.

Returns See [Error Codes](#)

int **wally_tx_input_init_alloc** (const unsigned char *txhash, size_t txhash_len, uint32_t utxo_index, uint32_t sequence, const unsigned char *script, size_t script_len, const struct wally_tx_witness_stack *witness, struct wally_tx_input **output)

Allocate and initialize a new transaction input.

Parameters

- **txhash** – The transaction hash of the transaction this input comes from.
- **txhash_len** – Size of txhash in bytes. Must be `WALLY_TXHASH_LEN`.
- **utxo_index** – The zero-based index of the transaction output in txhash that this input comes from.
- **sequence** – The sequence number for the input.
- **script** – The scriptSig for the input.
- **script_len** – Size of script in bytes.
- **witness** – The witness stack for the input, or NULL if no witness is present.

- **output** – Destination for the resulting transaction input.

Returns See [Error Codes](#)

int **wally_tx_input_free** (struct wally_tx_input *input)

Free a transaction input allocated by [wally_tx_input_init_alloc](#).

Parameters

- **input** – The transaction input to free.

Returns See [Error Codes](#)

int **wally_tx_output_init** (uint64_t *satoshi*, const unsigned char **script*, size_t *script_len*, struct wally_tx_output **output*)

Initialize a new transaction output.

Parameters

- **satoshi** – The amount of the output in satoshi.
- **script** – The scriptPubkey for the output.
- **script_len** – Size of *script* in bytes.
- **output** – Transaction output to initialize.

Returns See [Error Codes](#)

int **wally_tx_output_init_alloc** (uint64_t *satoshi*, const unsigned char **script*, size_t *script_len*, struct wally_tx_output ***output*)

Allocate and initialize a new transaction output.

Parameters

- **satoshi** – The amount of the output in satoshi.
- **script** – The scriptPubkey for the output.
- **script_len** – Size of *script* in bytes.
- **output** – Destination for the resulting transaction output.

Returns See [Error Codes](#)

int **wally_tx_output_clone_alloc** (const struct wally_tx_output **tx_output_in*, struct wally_tx_output ***output*)

Create a new copy of a transaction output.

Parameters

- **tx_output_in** – The transaction output to clone.
- **output** – Destination for the resulting transaction output copy.

Returns See [Error Codes](#)

int **wally_tx_output_clone** (const struct wally_tx_output **tx_output_in*, struct wally_tx_output **output*)

Create a new copy of a transaction output in place.

Parameters

- **tx_output_in** – The transaction output to clone.
- **output** – Destination for the resulting transaction output copy.

Note: `output` is overwritten in place, and not cleared first.

Returns See *Error Codes*

int **wally_tx_output_free** (struct wally_tx_output **output*)
Free a transaction output allocated by *wally_tx_output_init_alloc*.

Parameters

- **output** – The transaction output to free.

Returns See *Error Codes*

int **wally_tx_init_alloc** (uint32_t *version*, uint32_t *locktime*, size_t *inputs_allocation_len*, size_t *outputs_allocation_len*, struct wally_tx ***output*)
Allocate and initialize a new transaction.

Parameters

- **version** – The version of the transaction.
- **locktime** – The locktime of the transaction.
- **inputs_allocation_len** – The number of inputs to pre-allocate space for.
- **outputs_allocation_len** – The number of outputs to pre-allocate space for.
- **output** – Destination for the resulting transaction output.

Returns See *Error Codes*

int **wally_tx_clone_alloc** (const struct wally_tx **tx*, uint32_t *flags*, struct wally_tx ***output*)
Create a new copy of a transaction.

Parameters

- **tx** – The transaction to clone.
- **flags** – Flags controlling transaction creation. Must be 0.
- **output** – Destination for the resulting transaction copy.

Returns See *Error Codes*

int **wally_tx_add_input** (struct wally_tx **tx*, const struct wally_tx_input **input*)
Add a transaction input to a transaction.

Parameters

- **tx** – The transaction to add the input to.
- **input** – The transaction input to add to *tx*.

Returns See *Error Codes*

int **wally_tx_add_input_at** (struct wally_tx **tx*, uint32_t *index*, const struct wally_tx_input **input*)
Add a transaction input to a transaction at a given position.

Parameters

- **tx** – The transaction to add the input to.
- **index** – The zero-based index of the position to add the input at.
- **input** – The transaction input to add to *tx*.

Returns See *Error Codes*

```
int wally_tx_add_raw_input (struct wally_tx *tx, const unsigned char *txhash, size_t txhash_len,
                           uint32_t utxo_index, uint32_t sequence, const unsigned char *script,
                           size_t script_len, const struct wally_tx_witness_stack *witness,
                           uint32_t flags)
```

Add a transaction input to a transaction.

Parameters

- **tx** – The transaction to add the input to.
- **txhash** – The transaction hash of the transaction this input comes from.
- **txhash_len** – Size of txhash in bytes. Must be WALLY_TXHASH_LEN.
- **utxo_index** – The zero-based index of the transaction output in txhash that this input comes from.
- **sequence** – The sequence number for the input.
- **script** – The scriptSig for the input.
- **script_len** – Size of script in bytes.
- **witness** – The witness stack for the input, or NULL if no witness is present.
- **flags** – Flags controlling input creation. Must be 0.

Returns See *Error Codes*

```
int wally_tx_add_raw_input_at (struct wally_tx *tx, uint32_t index, const unsigned char *tx-
                               hash, size_t txhash_len, uint32_t utxo_index, uint32_t sequence,
                               const unsigned char *script, size_t script_len, const struct
                               wally_tx_witness_stack *witness, uint32_t flags)
```

Add a transaction input to a transaction in a given position.

Parameters

- **tx** – The transaction to add the input to.
- **index** – The zero-based index of the position to add the input at.
- **txhash** – The transaction hash of the transaction this input comes from.
- **txhash_len** – Size of txhash in bytes. Must be WALLY_TXHASH_LEN.
- **utxo_index** – The zero-based index of the transaction output in txhash that this input comes from.
- **sequence** – The sequence number for the input.
- **script** – The scriptSig for the input.
- **script_len** – Size of script in bytes.
- **witness** – The witness stack for the input, or NULL if no witness is present.
- **flags** – Flags controlling input creation. Must be 0.

Returns See *Error Codes*

```
int wally_tx_remove_input (struct wally_tx *tx, size_t index)
```

Remove a transaction input from a transaction.

Parameters

- **tx** – The transaction to remove the input from.

- **index** – The zero-based index of the input to remove.

Returns See *Error Codes*

int **wally_tx_set_input_script** (const struct wally_tx *tx, size_t index, const unsigned char *script, size_t script_len)

Set the scriptsig for an input in a transaction.

Parameters

- **tx** – The transaction to operate on.
- **index** – The zero-based index of the input to set the script on.
- **script** – The scriptSig for the input.
- **script_len** – Size of script in bytes.

Returns See *Error Codes*

int **wally_tx_set_input_witness** (const struct wally_tx *tx, size_t index, const struct wally_tx_witness_stack *stack)

Set the witness stack for an input in a transaction.

Parameters

- **tx** – The transaction to operate on.
- **index** – The zero-based index of the input to set the witness stack on.
- **stack** – The transaction witness stack to set.

Returns See *Error Codes*

int **wally_tx_add_output** (struct wally_tx *tx, const struct wally_tx_output *output)

Add a transaction output to a transaction.

Parameters

- **tx** – The transaction to add the output to.
- **output** – The transaction output to add to tx.

Returns See *Error Codes*

int **wally_tx_add_output_at** (struct wally_tx *tx, uint32_t index, const struct wally_tx_output *output)

Add a transaction output to a transaction at a given position.

Parameters

- **tx** – The transaction to add the output to.
- **index** – The zero-based index of the position to add the output at.
- **output** – The transaction output to add to tx.

Returns See *Error Codes*

int **wally_tx_add_raw_output** (struct wally_tx *tx, uint64_t satoshi, const unsigned char *script, size_t script_len, uint32_t flags)

Add a transaction output to a transaction.

Parameters

- **tx** – The transaction to add the output to.
- **satoshi** – The amount of the output in satoshi.
- **script** – The scriptPubkey for the output.

- **script_len** – Size of `script` in bytes.
- **flags** – Flags controlling output creation. Must be 0.

Returns See [Error Codes](#)

int **wally_tx_add_raw_output_at** (struct wally_tx *tx, uint32_t index, uint64_t satoshi, const unsigned char *script, size_t script_len, uint32_t flags)

Add a transaction output to a transaction at a given position.

Parameters

- **tx** – The transaction to add the output to.
- **index** – The zero-based index of the position to add the output at.
- **satoshi** – The amount of the output in satoshi.
- **script** – The scriptPubkey for the output.
- **script_len** – Size of `script` in bytes.
- **flags** – Flags controlling output creation. Must be 0.

Returns See [Error Codes](#)

int **wally_tx_remove_output** (struct wally_tx *tx, size_t index)

Remove a transaction output from a transaction.

Parameters

- **tx** – The transaction to remove the output from.
- **index** – The zero-based index of the output to remove.

Returns See [Error Codes](#)

int **wally_tx_get_witness_count** (const struct wally_tx *tx, size_t *written)

Get the number of inputs in a transaction that have witness data.

Parameters

- **tx** – The transaction to get the witnesses count from.
- **written** – Destination for the number of witness-containing inputs.

Returns See [Error Codes](#)

int **wally_tx_free** (struct wally_tx *tx)

Free a transaction allocated by `wally_tx_init_alloc`.

Parameters

- **tx** – The transaction to free.

Returns See [Error Codes](#)

int **wally_tx_get_txid** (const struct wally_tx *tx, unsigned char *bytes_out, size_t len)

Return the txid of a transaction.

Parameters

- **tx** – The transaction to compute the txid of.
- **bytes_out** – Destination for the txid.
- **len** – Size of `bytes_out`. Must be `WALLY_TXHASH_LEN`.

Note: The txid is expensive to compute.

Returns See [Error Codes](#)

int **wally_tx_get_length** (const struct wally_tx *tx, uint32_t flags, size_t *written)
Return the length of transaction once serialized into bytes.

Parameters

- **tx** – The transaction to find the serialized length of.
- **flags** – WALLY_TX_FLAG_ Flags controlling serialization options.
- **written** – Destination for the length of the serialized bytes.

Returns See [Error Codes](#)

int **wally_tx_from_bytes** (const unsigned char *bytes, size_t bytes_len, uint32_t flags, struct wally_tx **output)
Create a transaction from its serialized bytes.

Parameters

- **bytes** – Bytes to create the transaction from.
- **bytes_len** – Length of bytes in bytes.
- **flags** – WALLY_TX_FLAG_ Flags controlling serialization options.
- **output** – Destination for the resulting transaction.

Returns See [Error Codes](#)

int **wally_tx_from_hex** (const char *hex, uint32_t flags, struct wally_tx **output)
Create a transaction from its serialized bytes in hexadecimal.

Parameters

- **hex** – Hexadecimal string containing the transaction.
- **flags** – WALLY_TX_FLAG_ Flags controlling serialization options.
- **output** – Destination for the resulting transaction.

Returns See [Error Codes](#)

int **wally_tx_to_bytes** (const struct wally_tx *tx, uint32_t flags, unsigned char *bytes_out, size_t len, size_t *written)
Serialize a transaction to bytes.

Parameters

- **tx** – The transaction to serialize.
- **flags** – WALLY_TX_FLAG_ Flags controlling serialization options.
- **bytes_out** – Destination for the serialized transaction.
- **len** – Size of bytes_out in bytes.
- **written** – Destination for the length of the serialized transaction.

Returns See [Variable Length Output Buffers](#)

int **wally_tx_to_hex** (const struct wally_tx *tx, uint32_t flags, char **output)
Serialize a transaction to hex.

Parameters

- **tx** – The transaction to serialize.
- **flags** – `WALLY_TX_FLAG_` Flags controlling serialization options.
- **output** – Destination for the resulting hexadecimal string.

Note: The string returned should be freed using `wally_free_string`.

Returns See [Error Codes](#)

int **wally_tx_get_weight** (const struct wally_tx *tx, size_t *written)

Get the weight of a transaction.

Parameters

- **tx** – The transaction to get the weight of.
- **written** – Destination for the weight.

Returns See [Error Codes](#)

int **wally_tx_get_vsize** (const struct wally_tx *tx, size_t *written)

Get the virtual size of a transaction.

Parameters

- **tx** – The transaction to get the virtual size of.
- **written** – Destination for the virtual size.

Returns See [Error Codes](#)

int **wally_tx_vsize_from_weight** (size_t weight, size_t *written)

Compute transaction vsize from transaction weight.

Parameters

- **weight** – The weight to convert to a virtual size.
- **written** – Destination for the virtual size.

Returns See [Error Codes](#)

int **wally_tx_get_total_output_satoshi** (const struct wally_tx *tx, uint64_t *value_out)

Compute the total sum of all outputs in a transaction.

Parameters

- **tx** – The transaction to compute the total from.
- **value_out** – Destination for the output total.

Returns See [Error Codes](#)

int **wally_tx_get_btc_signature_hash** (const struct wally_tx *tx, size_t index, const unsigned char *script, size_t script_len, uint64_t satoshi, uint32_t sighash, uint32_t flags, unsigned char *bytes_out, size_t len)

Create a BTC transaction for signing and return its hash.

Parameters

- **tx** – The transaction to generate the signature hash from.

- **index** – The input index of the input being signed for.
- **script** – The (unprefixed) scriptCode for the input being signed.
- **script_len** – Size of `script` in bytes.
- **satoshi** – The amount spent by the input being signed for. Only used if flags includes `WALLY_TX_FLAG_USE_WITNESS`, pass 0 otherwise.
- **sighash** – `WALLY_SIGHASH_` flags specifying the type of signature desired.
- **flags** – `WALLY_TX_FLAG_USE_WITNESS` to generate a BIP 143 signature, or 0 to generate a pre-segwit Bitcoin signature.
- **bytes_out** – Destination for the signature hash.
- **len** – Size of `bytes_out`. Must be `SHA256_LEN`.

Returns See *Error Codes*

```
int wally_tx_get_btc_taproot_signature_hash(const struct wally_tx *tx, size_t index,
                                           const struct wally_map *scripts, const
                                           uint64_t *values, size_t num_values, const
                                           unsigned char *tapleaf_script, size_t ta-
                                           pleaf_script_len, uint32_t key_version,
                                           uint32_t codesep_position, const unsigned
                                           char *annex, size_t annex_len, uint32_t sighash,
                                           uint32_t flags, unsigned char *bytes_out,
                                           size_t len)
```

Create a BTC transaction for taproot signing and return its hash.

Parameters

- **tx** – The transaction to generate the signature hash from.
- **index** – The input index of the input being signed for.
- **scripts** – Map of input index to (unprefixed) scriptCodes for each input in `tx`.
- **values** – The value in satoshi for each input in `tx`.
- **num_values** – The number of elements in `values`.
- **tapleaf_script** – BIP342 tapscript being spent.
- **tapleaf_script_len** – Length of `tapleaf_script` in bytes.
- **key_version** – Version of pubkey in tapscript. Must be set to 0x00 or 0x01.
- **codesep_position** – BIP342 codeseparator position or `WALLY_NO_CODESEPARATOR` if none.
- **annex** – BIP341 annex, or NULL if none.
- **annex_len** – Length of annex in bytes.
- **sighash** – `WALLY_SIGHASH_` flags specifying the type of signature desired.
- **flags** – Flags controlling signature generation. Must be 0.
- **bytes_out** – Destination for the resulting signature hash.
- **len** – Size of `bytes_out`. Must be `SHA256_LEN`.

Returns See *Error Codes*

int **wally_tx_get_signature_hash** (const struct wally_tx *tx, size_t index, const unsigned char *script, size_t script_len, const unsigned char *extra, size_t extra_len, uint32_t extra_offset, uint64_t satoshi, uint32_t sighash, uint32_t tx_sighash, uint32_t flags, unsigned char *bytes_out, size_t len)

Create a transaction for signing and return its hash.

Parameters

- **tx** – The transaction to generate the signature hash from.
- **index** – The input index of the input being signed for.
- **script** – The (unprefixed) scriptCode for the input being signed.
- **script_len** – Size of script in bytes.
- **extra** – Extra bytes to include in the transaction preimage.
- **extra_len** – Size of extra in bytes.
- **extra_offset** – Offset with the preimage to store extra. To store it at the end of the preimage, use 0xffffffff.
- **satoshi** – The amount spent by the input being signed for. Only used if flags includes WALLY_TX_FLAG_USE_WITNESS, pass 0 otherwise.
- **sighash** – WALLY_SIGHASH_ flags specifying the type of signature desired.
- **tx_sighash** – The 32bit sighash value to include in the preimage to hash. This must be given in host CPU endianness; For normal Bitcoin signing the value of sighash should be given.
- **flags** – WALLY_TX_FLAG_USE_WITNESS to generate a BIP 143 signature, or 0 to generate a pre-segwit Bitcoin signature.
- **bytes_out** – Destination for the signature hash.
- **len** – Size of bytes_out. Must be SHA256_LEN.

Returns See [Error Codes](#)

int **wally_tx_is_coinbase** (const struct wally_tx *tx, size_t *written)

Determine if a transaction is a coinbase transaction.

Parameters

- **tx** – The transaction to check.
- **written** – 1 if the transaction is a coinbase transaction, otherwise 0.

Returns See [Error Codes](#)

int **wally_tx_elements_input_issuance_set** (struct wally_tx_input *input, const unsigned char *nonce, size_t nonce_len, const unsigned char *entropy, size_t entropy_len, const unsigned char *issuance_amount, size_t issuance_amount_len, const unsigned char *inflation_keys, size_t inflation_keys_len, const unsigned char *issuance_amount_rangeproof, size_t issuance_amount_rangeproof_len, const unsigned char *inflation_keys_rangeproof, size_t inflation_keys_rangeproof_len)

Set issuance data on an input.

Parameters

- **input** – The input to add to.
- **nonce** – Asset issuance or revelation blinding factor.
- **nonce_len** – Size of nonce in bytes. Must be `WALLY_TX_ASSET_TAG_LEN`.
- **entropy** – Entropy for the asset tag calculation.
- **entropy_len** – Size of entropy in bytes. Must be `WALLY_TX_ASSET_TAG_LEN`.
- **issuance_amount** – The (blinded) issuance amount.
- **issuance_amount_len** – Size of issuance_amount in bytes.
- **inflation_keys** – The (blinded) token reissuance amount.
- **inflation_keys_len** – Size of inflation_keys in bytes.
- **issuance_amount_rangeproof** – Issuance amount rangeproof.
- **issuance_amount_rangeproof_len** – Size of issuance_amount_rangeproof in bytes.
- **inflation_keys_rangeproof** – Inflation keys rangeproof.
- **inflation_keys_rangeproof_len** – Size of inflation_keys_rangeproof in bytes.

Returns See *Error Codes*

int **wally_tx_elements_input_issuance_free** (struct wally_tx_input *input)
Free issuance data on an input.

Parameters

- **input** – The input issuance data to free.

Returns See *Error Codes*

int **wally_tx_elements_input_init_alloc** (const unsigned char *txhash, size_t txhash_len, uint32_t utxo_index, uint32_t sequence, const unsigned char *script, size_t script_len, const struct wally_tx_witness_stack *witness, const unsigned char *nonce, size_t nonce_len, const unsigned char *issuance_amount, size_t issuance_amount_len, const unsigned char *inflation_keys, size_t inflation_keys_len, const unsigned char *issuance_amount_rangeproof, size_t issuance_amount_rangeproof_len, const unsigned char *inflation_keys_rangeproof, size_t inflation_keys_rangeproof_len, const struct wally_tx_witness_stack *pegin_witness, struct wally_tx_input **output)

Allocate and initialize a new elements transaction input.

Parameters

- **txhash** – The transaction hash of the transaction this input comes from.
- **txhash_len** – Size of txhash in bytes. Must be `WALLY_TXHASH_LEN`.
- **utxo_index** – The zero-based index of the transaction output in txhash that this input comes from.

- **sequence** – The sequence number for the input.
- **script** – The scriptSig for the input.
- **script_len** – Size of `script` in bytes.
- **witness** – The witness stack for the input, or NULL if no witness is present.
- **nonce** – Asset issuance or revelation blinding factor.
- **nonce_len** – Size of `nonce` in bytes. Must be `WALLY_TX_ASSET_TAG_LEN`.
- **entropy** – Entropy for the asset tag calculation.
- **entropy_len** – Size of `entropy` in bytes. Must be `WALLY_TX_ASSET_TAG_LEN`.
- **issuance_amount** – The (blinded) issuance amount.
- **issuance_amount_len** – Size of `issuance_amount` in bytes.
- **inflation_keys** – The (blinded) token reissuance amount.
- **inflation_keys_len** – Size of `inflation_keys` in bytes.
- **issuance_amount_rangeproof** – Issuance amount rangeproof.
- **issuance_amount_rangeproof_len** – Size of `issuance_amount_rangeproof` in bytes.
- **inflation_keys_rangeproof** – Inflation keys rangeproof.
- **inflation_keys_rangeproof_len** – Size of `inflation_keys_rangeproof` in bytes.
- **pegin_witness** – The pegin witness stack for the input, or NULL if no witness is present.
- **output** – Destination for the resulting transaction input.

Returns See [Error Codes](#)

int **wally_tx_elements_input_is_pegin** (const struct wally_tx_input *input, size_t *written)
Determine if an input is a pegin.

Parameters

- **input** – The input to check.
- **written** – 1 if the input is a pegin, otherwise 0.

Returns See [Error Codes](#)

int **wally_tx_elements_output_commitment_set** (struct wally_tx_output *output, const unsigned char *asset, size_t asset_len, const unsigned char *value, size_t value_len, const unsigned char *nonce, size_t nonce_len, const unsigned char *surjectionproof, size_t surjectionproof_len, const unsigned char *rangeproof, size_t rangeproof_len)

Set commitment data on an output.

Parameters

- **output** – The output to add to.
- **asset** – The commitment to a possibly blinded asset.
- **asset_len** – Size of `asset` in bytes. Must be `WALLY_TX_ASSET_CT_ASSET_LEN`.

- **value** – The commitment to a possibly blinded value.
- **value_len** – Size of **value** in bytes. Must be `WALLY_TX_ASSET_CT_VALUE_LEN` or `WALLY_TX_ASSET_CT_VALUE_UNBLIND_LEN`.
- **nonce** – The commitment used to create the nonce (with the blinding key) for the range proof.
- **nonce_len** – Size of **nonce** in bytes. Must be `WALLY_TX_ASSET_CT_NONCE_LEN`.
- **surjectionproof** – surjection proof.
- **surjectionproof_len** – Size of **surjectionproof** in bytes.
- **rangeproof** – rangeproof.
- **rangeproof_len** – Size of **rangeproof** in bytes.

Returns See *Error Codes*

int **wally_tx_elements_output_commitment_free** (struct wally_tx_output **output*)
Free commitment data on an output.

Parameters

- **output** – The output with the commitment data to free.

Returns See *Error Codes*

int **wally_tx_elements_output_init** (const unsigned char **script*, size_t *script_len*, const unsigned char **asset*, size_t *asset_len*, const unsigned char **value*, size_t *value_len*, const unsigned char **nonce*, size_t *nonce_len*, const unsigned char **surjectionproof*, size_t *surjectionproof_len*, const unsigned char **rangeproof*, size_t *rangeproof_len*, struct wally_tx_output **output*)
Initialize a new elements transaction output in place.

Parameters

- **script** – The scriptPubkey for the output.
- **script_len** – Size of **script** in bytes.
- **asset** – The asset tag of the output.
- **asset_len** – Size of **asset** in bytes. Must be `WALLY_TX_ASSET_CT_ASSET_LEN`.
- **value** – The commitment to a possibly blinded value.
- **value_len** – Size of **value** in bytes. Must be `WALLY_TX_ASSET_CT_VALUE_LEN` or `WALLY_TX_ASSET_CT_VALUE_UNBLIND_LEN`.
- **nonce** – The commitment used to create the nonce (with the blinding key) for the range proof.
- **nonce_len** – Size of **nonce** in bytes. Must be `WALLY_TX_ASSET_CT_NONCE_LEN`.
- **surjectionproof** – The surjection proof.
- **surjectionproof_len** – Size of **surjectionproof** in bytes.
- **rangeproof** – The range proof.
- **rangeproof_len** – Size of **rangeproof** in bytes.
- **output** – Destination for the resulting transaction output copy.

Note: `output` is overwritten in place, and not cleared first.

Returns See [Error Codes](#)

```
int wally_tx_elements_output_init_alloc (const unsigned char *script, size_t script_len, const
                                         unsigned char *asset, size_t asset_len, const un-
                                         signed char *value, size_t value_len, const un-
                                         signed char *nonce, size_t nonce_len, const unsigned
                                         char *surjectionproof, size_t surjectionproof_len,
                                         const unsigned char *rangeproof, size_t range-
                                         proof_len, struct wally_tx_output **output)
```

Allocate and initialize a new elements transaction output.

Parameters

- **script** – The scriptPubkey for the output.
- **script_len** – Size of `script` in bytes.
- **asset** – The asset tag of the output.
- **asset_len** – Size of `asset` in bytes. Must be `WALLY_TX_ASSET_CT_ASSET_LEN`.
- **value** – The commitment to a possibly blinded value.
- **value_len** – Size of `value` in bytes. Must be `WALLY_TX_ASSET_CT_VALUE_LEN` or `WALLY_TX_ASSET_CT_VALUE_UNBLIND_LEN`.
- **nonce** – The commitment used to create the nonce (with the blinding key) for the range proof.
- **nonce_len** – Size of `nonce` in bytes. Must be `WALLY_TX_ASSET_CT_NONCE_LEN`.
- **surjectionproof** – The surjection proof.
- **surjectionproof_len** – Size of `surjectionproof` in bytes.
- **rangeproof** – The range proof.
- **rangeproof_len** – Size of `rangeproof` in bytes.
- **output** – Destination for the resulting transaction output.

Returns See [Error Codes](#)

```
int wally_tx_add_elements_raw_input (struct wally_tx *tx, const unsigned char *txhash,
                                     size_t txhash_len, uint32_t utxo_index, uint32_t se-
                                     quence, const unsigned char *script, size_t script_len,
                                     const struct wally_tx_witness_stack *witness, const un-
                                     signed char *nonce, size_t nonce_len, const unsigned
                                     char *entropy, size_t entropy_len, const unsigned char *is-
                                     suance_amount, size_t issuance_amount_len, const
                                     unsigned char *inflation_keys, size_t inflation_keys_len,
                                     const unsigned char *issuance_amount_rangeproof,
                                     size_t issuance_amount_rangeproof_len, const
                                     unsigned char *inflation_keys_rangeproof,
                                     size_t inflation_keys_rangeproof_len, const struct
                                     wally_tx_witness_stack *pegin_witness, uint32_t flags)
```

Add an elements transaction input to a transaction.

Parameters

- **tx** – The transaction to add the input to.
- **txhash** – The transaction hash of the transaction this input comes from.
- **txhash_len** – Size of txhash in bytes. Must be WALLY_TXHASH_LEN.
- **utxo_index** – The zero-based index of the transaction output in txhash that this input comes from.
- **sequence** – The sequence number for the input.
- **script** – The scriptSig for the input.
- **script_len** – Size of script in bytes.
- **witness** – The witness stack for the input, or NULL if no witness is present.
- **nonce** – Asset issuance or revelation blinding factor.
- **nonce_len** – Size of nonce in bytes. Must be WALLY_TX_ASSET_TAG_LEN.
- **entropy** – Entropy for the asset tag calculation.
- **entropy_len** – Size of entropy in bytes. Must be WALLY_TX_ASSET_TAG_LEN.
- **issuance_amount** – The (blinded) issuance amount.
- **issuance_amount_len** – Size of issuance_amount in bytes.
- **inflation_keys** – The (blinded) token reissuance amount.
- **inflation_keys_len** – Size of inflation_keys in bytes.
- **issuance_amount_rangeproof** – Issuance amount rangeproof.
- **issuance_amount_rangeproof_len** – Size of issuance_amount_rangeproof in bytes.
- **inflation_keys_rangeproof** – Inflation keys rangeproof.
- **inflation_keys_rangeproof_len** – Size of inflation_keys_rangeproof in bytes.
- **pegin_witness** – The pegin witness stack for the input, or NULL if no witness is present.
- **flags** – Flags controlling input creation. Must be 0.

Returns See [Error Codes](#)

```
int wally_tx_add_elements_raw_input_at (struct wally_tx *tx, uint32_t index, const
    unsigned char *txhash, size_t txhash_len,
    uint32_t utxo_index, uint32_t sequence, const un-
    signed char *script, size_t script_len, const struct
    wally_tx_witness_stack *witness, const unsigned
    char *nonce, size_t nonce_len, const unsigned char *en-
    tropy, size_t entropy_len, const unsigned char *is-
    suance_amount, size_t issuance_amount_len, const un-
    signed char *inflation_keys, size_t inflation_keys_len,
    const unsigned char *issuance_amount_rangeproof,
    size_t issuance_amount_rangeproof_len, const
    unsigned char *inflation_keys_rangeproof,
    size_t inflation_keys_rangeproof_len, const
    struct wally_tx_witness_stack *pegin_witness,
    uint32_t flags)
```

Add an elements transaction input to a transaction at a given position.

Parameters

- **tx** – The transaction to add the input to.
- **index** – The zero-based index of the position to add the input at.
- **txhash** – The transaction hash of the transaction this input comes from.
- **txhash_len** – Size of txhash in bytes. Must be WALLY_TXHASH_LEN.
- **utxo_index** – The zero-based index of the transaction output in txhash that this input comes from.
- **sequence** – The sequence number for the input.
- **script** – The scriptSig for the input.
- **script_len** – Size of script in bytes.
- **witness** – The witness stack for the input, or NULL if no witness is present.
- **nonce** – Asset issuance or revelation blinding factor.
- **nonce_len** – Size of nonce in bytes. Must be WALLY_TX_ASSET_TAG_LEN.
- **entropy** – Entropy for the asset tag calculation.
- **entropy_len** – Size of entropy in bytes. Must be WALLY_TX_ASSET_TAG_LEN.
- **issuance_amount** – The (blinded) issuance amount.
- **issuance_amount_len** – Size of issuance_amount in bytes.
- **inflation_keys** – The (blinded) token reissuance amount.
- **inflation_keys_len** – Size of inflation_keys in bytes.
- **issuance_amount_rangeproof** – Issuance amount rangeproof.
- **issuance_amount_rangeproof_len** – Size of issuance_amount_rangeproof in bytes.
- **inflation_keys_rangeproof** – Inflation keys rangeproof.
- **inflation_keys_rangeproof_len** – Size of inflation_keys_rangeproof in bytes.
- **pegin_witness** – The pegin witness stack for the input, or NULL if no witness is present.
- **flags** – Flags controlling input creation. Must be 0.

Returns See *Error Codes*

```
int wally_tx_add_elements_raw_output (struct wally_tx *tx, const unsigned char *script,
                                     size_t script_len, const unsigned char *asset, size_t asset_len,
                                     const unsigned char *value, size_t value_len,
                                     const unsigned char *nonce, size_t nonce_len, const unsigned char *surjectionproof,
                                     size_t surjectionproof_len, const unsigned char *rangeproof, size_t rangeproof_len,
                                     uint32_t flags)
```

Add a elements transaction output to a transaction.

Parameters

- **tx** – The transaction to add the output to.
- **script** – The scriptPubkey for the output.

- **script_len** – Size of `script` in bytes.
- **asset** – The asset tag of the output.
- **asset_len** – Size of `asset` in bytes. Must be `WALLY_TX_ASSET_CT_ASSET_LEN`.
- **value** – The commitment to a possibly blinded value.
- **value_len** – Size of `value` in bytes. Must be `WALLY_TX_ASSET_CT_VALUE_LEN` or `WALLY_TX_ASSET_CT_VALUE_UNBLIND_LEN`.
- **nonce** – The commitment used to create the nonce (with the blinding key) for the range proof.
- **nonce_len** – Size of `nonce` in bytes. Must be `WALLY_TX_ASSET_CT_NONCE_LEN`.
- **surjectionproof** – The surjection proof.
- **surjectionproof_len** – Size of `surjectionproof` in bytes.
- **rangeproof** – The range proof.
- **rangeproof_len** – Size of `rangeproof` in bytes.
- **flags** – Flags controlling output creation. Must be 0.

Returns See [Error Codes](#)

```
int wally_tx_add_elements_raw_output_at (struct wally_tx *tx, uint32_t index, const unsigned char *script, size_t script_len, const unsigned char *asset, size_t asset_len, const unsigned char *value, size_t value_len, const unsigned char *nonce, size_t nonce_len, const unsigned char *surjectionproof, size_t surjectionproof_len, const unsigned char *rangeproof, size_t rangeproof_len, uint32_t flags)
```

Add a elements transaction output to a transaction at a given position.

Parameters

- **tx** – The transaction to add the output to.
- **index** – The zero-based index of the position to add the output at.
- **script** – The scriptPubkey for the output.
- **script_len** – Size of `script` in bytes.
- **asset** – The asset tag of the output.
- **asset_len** – Size of `asset` in bytes. Must be `WALLY_TX_ASSET_CT_ASSET_LEN`.
- **value** – The commitment to a possibly blinded value.
- **value_len** – Size of `value` in bytes. Must be `WALLY_TX_ASSET_CT_VALUE_LEN` or `WALLY_TX_ASSET_CT_VALUE_UNBLIND_LEN`.
- **nonce** – The commitment used to create the nonce (with the blinding key) for the range proof.
- **nonce_len** – Size of `nonce` in bytes. Must be `WALLY_TX_ASSET_CT_NONCE_LEN`.
- **surjectionproof** – The surjection proof.
- **surjectionproof_len** – Size of `surjectionproof` in bytes.
- **rangeproof** – The range proof.

- **rangeproof_len** – Size of rangeproof in bytes.
- **flags** – Flags controlling output creation. Must be 0.

Returns See *Error Codes*

int **wally_tx_is_elements** (const struct wally_tx *tx, size_t *written)
Determine if a transaction is an elements transaction.

Parameters

- **tx** – The transaction to check.
- **written** – 1 if the transaction is an elements transaction, otherwise 0.

Returns See *Error Codes*

int **wally_tx_confidential_value_from_satoshi** (uint64_t satoshi, unsigned char *bytes_out,
size_t len)
Convert satoshi to an explicit confidential value representation.

Parameters

- **satoshi** – The value in satoshi to convert.
- **bytes_out** – Destination for the confidential value bytes.
- **len** – Size of bytes_out. Must be WALLY_TX_ASSET_CT_VALUE_UNBLIND_LEN.

Returns See *Error Codes*

int **wally_tx_confidential_value_to_satoshi** (const unsigned char *value, size_t value_len,
uint64_t *value_out)
Convert an explicit confidential value representation to satoshi.

Parameters

- **value** – The confidential value bytes.
- **value_len** – Size of value in bytes. Must be WALLY_TX_ASSET_CT_VALUE_UNBLIND_LEN.
- **value_out** – The converted value in satoshi.

Returns See *Error Codes*

int **wally_tx_get_elements_signature_hash** (const struct wally_tx *tx, size_t index, const unsigned char *script, size_t script_len, const unsigned char *value, size_t value_len, uint32_t sighash, uint32_t flags, unsigned char *bytes_out, size_t len)
Create an Elements transaction for signing and return its hash.

Parameters

- **tx** – The transaction to generate the signature hash from.
- **index** – The input index of the input being signed for.
- **script** – The (unprefixed) scriptCode for the input being signed.
- **script_len** – Size of script in bytes.
- **value** – The (confidential) value spent by the input being signed for. Only used if flags includes WALLY_TX_FLAG_USE_WITNESS, pass NULL otherwise.
- **value_len** – Size of value in bytes.
- **sighash** – WALLY_SIGHASH_ flags specifying the type of signature desired.

- **flags** – `WALLY_TX_FLAG_USE_WITNESS` to generate a BIP 143 signature, or 0 to generate a pre-segwit Bitcoin signature.
- **bytes_out** – Destination for the signature hash.
- **len** – Size of `bytes_out`. Must be `SHA256_LEN`.

Returns See [Error Codes](#)

int **wally_tx_elements_issuance_generate_entropy** (const unsigned char *txhash, size_t txhash_len, uint32_t utxo_index, const unsigned char *contract_hash, size_t contract_hash_len, unsigned char *bytes_out, size_t len)

Calculate the asset entropy from a prevout and the Ricardian contract hash.

Parameters

- **txhash** – The prevout transaction hash.
- **txhash_len** – Size of `txhash` in bytes. Must be `WALLY_TXHASH_LEN`.
- **utxo_index** – The zero-based index of the transaction output in `txhash` to use.
- **contract_hash** – The issuer specified Ricardian contract hash.
- **contract_hash_len** – Size of `contract_hash` in bytes. Must be `SHA256_LEN`.
- **bytes_out** – Destination for the asset entropy.
- **len** – Size of `bytes_out`. Must be `SHA256_LEN`.

Returns See [Error Codes](#)

int **wally_tx_elements_issuance_calculate_asset** (const unsigned char *entropy, size_t entropy_len, unsigned char *bytes_out, size_t len)

Calculate the asset from the entropy.

Parameters

- **entropy** – The asset entropy.
- **entropy_len** – Size of `entropy` in bytes. Must be `SHA256_LEN`.
- **bytes_out** – Destination for the asset tag.
- **len** – Size of `bytes_out`. Must be `SHA256_LEN`.

Returns See [Error Codes](#)

int **wally_tx_elements_issuance_calculate_reissuance_token** (const unsigned char *entropy, size_t entropy_len, uint32_t flags, unsigned char *bytes_out, size_t len)

Calculate a re-issuance token from an asset's entropy.

Parameters

- **entropy** – The asset entropy.
- **entropy_len** – Size of `entropy` in bytes. Must be `SHA256_LEN`.
- **flags** – `WALLY_TX_FLAG_BLINDED_INITIAL_ISSUANCE` if initial issuance was blinded, pass 0 otherwise.
- **bytes_out** – Destination for the re-issuance token.

- **len** – Size of `bytes_out`. Must be `SHA256_LEN`.

Returns See *Error Codes*

Elements Functions

int **wally_asset_generator_from_bytes** (const unsigned char *asset, size_t asset_len, const unsigned char *abf, size_t abf_len, unsigned char *bytes_out, size_t len)

Create an Asset Generator from an either an asset commitment or asset tag plus blinding factor.

Parameters

- **asset** – Asset Commitment or Tag to create a generator for.
- **asset_len** – Length of asset in bytes. Must be ASSET_COMMITMENT_LEN or ASSET_TAG_LEN.
- **abf** – Asset Blinding Factor (Random entropy to blind with). Must be NULL when asset is a commitment.
- **abf_len** – Length of abf in bytes. Must be BLINDING_FACTOR_LEN if abf is non-NULL.
- **bytes_out** – Destination for the resulting Asset Generator.
- **len** – Size of bytes_out. Must be ASSET_GENERATOR_LEN.

Returns See *Error Codes*

int **wally_ecdh_nonce_hash** (const unsigned char *pub_key, size_t pub_key_len, const unsigned char *priv_key, size_t priv_key_len, unsigned char *bytes_out, size_t len)

Generate a rangeproof nonce hash via SHA256(ECDH(pub_key, priv_key)).

Parameters

- **pub_key** – Public blinding key.
- **pub_key_len** – Length of pub_key in bytes. Must be EC_PUBLIC_KEY_LEN
- **priv_key** – Ephemeral (randomly generated) private key.
- **priv_key_len** – Length of priv_key in bytes. Must be EC_PRIVATE_KEY_LEN.
- **bytes_out** – Destination for the resulting nonce hash.
- **len** – Size of bytes_out. Must be SHA256_LEN.

Note: The public blinding key can be retrieved from a confidential address using `wally_confidential_addr_to_ec_public_key`. If `priv_key` is invalid, then `WALLY_ERROR` is returned.

Note: The computation can also be performed with the private key corresponding to `pub_key` and the public key corresponding to `priv_key` giving the same result.

Returns See *Error Codes*

int **wally_asset_final_vbf** (const uint64_t **values*, size_t *num_values*, size_t *num_inputs*, const unsigned char **abf*, size_t *abf_len*, const unsigned char **vbf*, size_t *vbf_len*, unsigned char **bytes_out*, size_t *len*)

Generate the final value blinding factor required for blinding a confidential transaction.

Parameters

- **values** – Array of values in satoshi
- **num_values** – Length of *values*, also the number of elements in all three of the input arrays, which is equal to *num_inputs* plus the number of outputs.
- **num_inputs** – Number of elements in the input arrays that represent inputs. The number of outputs is implicitly *num_values* - *num_inputs*.
- **abf** – Array of bytes representing *num_values* asset blinding factors.
- **abf_len** – Length of *abf* in bytes. Must be *num_values* * `BLINDING_FACTOR_LEN`.
- **vbf** – Array of bytes representing (*num_values* - 1) value blinding factors.
- **vbf_len** – Length of *vbf* in bytes. Must be (*num_values* - 1) * `BLINDING_FACTOR_LEN`.
- **bytes_out** – Buffer to receive the final value blinding factor.
- **len** – Size of *bytes_out*. Must be `BLINDING_FACTOR_LEN`.

Returns See *Error Codes*

int **wally_asset_scalar_offset** (uint64_t *value*, const unsigned char **abf*, size_t *abf_len*, const unsigned char **vbf*, size_t *vbf_len*, unsigned char **bytes_out*, size_t *len*)

Compute the scalar offset used for blinding a confidential transaction.

Parameters

- **value** – The value in satoshi.
- **abf** – Asset blinding factor.
- **abf_len** – Length of *abf*. Must be `BLINDING_FACTOR_LEN`.
- **vbf** – Value blinding factor.
- **vbf_len** – Length of *vbf*. Must be `BLINDING_FACTOR_LEN`.
- **bytes_out** – Destination to receive the scalar offset.
- **len** – Size of *bytes_out*. Must be `EC_SCALAR_LEN`.

Returns See *Error Codes*

```
int wally_asset_value_commitment(uint64_t value, const unsigned char *vbf, size_t vbf_len,
                                const unsigned char *generator, size_t generator_len, unsigned
                                char *bytes_out, size_t len)
```

Calculate a value commitment.

Parameters

- **value** – Output value in satoshi.
- **vbf** – Value Blinding Factor.
- **vbf_len** – Length of vbf. Must be `BLINDING_FACTOR_LEN`.
- **generator** – Asset generator from *wally_asset_generator_from_bytes*.
- **generator_len** – Length of generator. Must be `ASSET_GENERATOR_LEN`.
- **bytes_out** – Buffer to receive value commitment.
- **len** – Size of bytes_out. Must be `ASSET_COMMITMENT_LEN`.

Returns See *Error Codes*

```
int wally_asset_rangeproof_with_nonce(uint64_t value, const unsigned char *nonce_hash,
                                      size_t nonce_hash_len, const unsigned char *as-
                                      set, size_t asset_len, const unsigned char *abf,
                                      size_t abf_len, const unsigned char *vbf, size_t vbf_len,
                                      const unsigned char *commitment, size_t commit-
                                      ment_len, const unsigned char *extra, size_t extra_len,
                                      const unsigned char *generator, size_t generator_len,
                                      uint64_t min_value, int exp, int min_bits, unsigned
                                      char *bytes_out, size_t len, size_t *written)
```

Generate a rangeproof using a nonce.

Parameters

- **value** – Value of the output in satoshi.
- **nonce_hash** – Nonce for rangeproof generation, usually from *wally_ecdh_nonce_hash*.
- **nonce_hash_len** – Length of nonce_hash. Must be `SHA256_LEN`.
- **asset** – Asset id of output.
- **asset_len** – Length of asset. Must be `ASSET_TAG_LEN`.
- **abf** – Asset blinding factor. Randomly generated for each output.
- **abf_len** – Length of abf. Must be `BLINDING_FACTOR_LEN`.
- **vbf** – Value blinding factor. Randomly generated for each output except the last, which is generate by calling *wally_asset_final_vbf*.
- **vbf_len** – Length of vbf. Must be `BLINDING_FACTOR_LEN`.
- **commitment** – Value commitment from *wally_asset_value_commitment*.
- **commitment_len** – Length of commitment. Must be `ASSET_COMMITMENT_LEN`.
- **extra** – Set this to the script pubkey of the output.
- **extra_len** – Length of extra, i.e. script pubkey.
- **generator** – Asset generator from *wally_asset_generator_from_bytes*.

- **generator_len** – Length of generator`. Must be ``ASSET_GENERATOR_LEN.
- **min_value** – Recommended value 1.
- **exp** – Exponent value. $-1 \geq \text{exp} \geq 18$. Recommended value 0.
- **min_bits** – $0 \geq \text{min_bits} \geq 64$. Recommended value 52.
- **bytes_out** – Buffer to receive rangeproof.
- **n** – Size of bytes_out. Passing ASSET_RANGEPROOF_MAX_LEN will ensure the buffer is large enough.
- **written** – Number of bytes actually written to bytes_out.

Returns See *Variable Length Output Buffers*

```
int wally_asset_rangeproof (uint64_t value, const unsigned char *pub_key, size_t pub_key_len, const
                           unsigned char *priv_key, size_t priv_key_len, const unsigned char *asset,
                           size_t asset_len, const unsigned char *abf, size_t abf_len, const unsigned
                           char *vbf, size_t vbf_len, const unsigned char *commitment, size_t com-
                           mitment_len, const unsigned char *extra, size_t extra_len, const un-
                           signed char *generator, size_t generator_len, uint64_t min_value, int exp,
                           int min_bits, unsigned char *bytes_out, size_t len, size_t *written)
```

Generate a rangeproof.

This convenience function generates a nonce hash with *wally_ecdh_nonce_hash* and then calls *wally_asset_rangeproof_with_nonce*.

Parameters

- **n** – Size of bytes_out. Passing ASSET_RANGEPROOF_MAX_LEN will ensure the buffer is large enough.

Returns See *Variable Length Output Buffers*

```
int wally_explicit_rangeproof (uint64_t value, const unsigned char *nonce, size_t nonce_len, const
                              unsigned char *vbf, size_t vbf_len, const unsigned char *commitment,
                              size_t commitment_len, const unsigned char *generator, size_t gen-
                              erator_len, unsigned char *bytes_out, size_t len, size_t *written)
```

Generate an explicit value rangeproof.

The nonce for this function should be randomly generated. See *wally_asset_rangeproof_with_nonce*.

Parameters

- **n** – Size of bytes_out. Passing ASSET_EXPLICIT_RANGEPROOF_MAX_LEN will ensure the buffer is large enough.

Returns See *Variable Length Output Buffers*

```
int wally_explicit_rangeproof_verify (const unsigned char *rangeproof, size_t rangeproof_len,
                                     uint64_t value, const unsigned char *commitment,
                                     size_t commitment_len, const unsigned char *generator,
                                     size_t generator_len)
```

Verify an explicit value rangeproof proves a given value.

Parameters

- **rangeproof** – The explicit value rangeproof to validate.
- **rangeproof_len** – Length of rangeproof in bytes.

- **value** – The expected value that the explicit rangeproof proves.
- **commitment** – Value commitment from `wally_asset_value_commitment`.
- **commitment_len** – Length of commitment. Must be `ASSET_COMMITMENT_LEN`.
- **generator** – Asset generator from `wally_asset_generator_from_bytes`.
- **generator_len** – Length of generator`. Must be ```ASSET_GENERATOR_LEN`.

Returns See *Error Codes*

int **wally_asset_surjectionproof_size** (size_t *num_inputs*, size_t **written*)

Return the required buffer size for receiving a surjection proof

Parameters

- **num_inputs** – Number of inputs.
- **written** – Destination for the surjection proof size.

Returns See *Error Codes*

int **wally_asset_surjectionproof_len** (const unsigned char **output_asset*, size_t *output_asset_len*, const unsigned char **output_abf*, size_t *output_abf_len*, const unsigned char **output_generator*, size_t *output_generator_len*, const unsigned char **bytes*, size_t *bytes_len*, const unsigned char **asset*, size_t *asset_len*, const unsigned char **abf*, size_t *abf_len*, const unsigned char **generator*, size_t *generator_len*, size_t **written*)

Compute the length of an asset surjection proof.

Parameters

- **output_asset** – asset id for the output.
- **output_asset_len** – Length of asset. Must be `ASSET_TAG_LEN`.
- **output_abf** – Asset blinding factor for the output. Generated randomly for each output.
- **output_abf_len** – Length of output_abf. Must be `BLINDING_FACTOR_LEN`.
- **output_generator** – Asset generator from `wally_asset_generator_from_bytes`.
- **output_generator_len** – Length of output_generator`. Must be ```ASSET_GENERATOR_LEN`.
- **bytes** – Must be generated randomly for each output.
- **bytes_len** – Length of bytes. Must be 32.
- **asset** – Array of input asset tags.
- **asset_len** – Length of asset`. Must be ```ASSET_TAG_LEN` * number of inputs.
- **abf** – Array of input asset blinding factors.
- **abf_len** – Length of abf. Must be `BLINDING_FACTOR_LEN` * number of inputs.
- **generator** – Array of input asset generators.
- **generator_len** – Length of generator. Must be `ASSET_GENERATOR_LEN` * number of inputs.
- **written** – Number of bytes actually written to *bytes_out*.

Returns See *Error Codes*

int **wally_asset_surjectionproof** (const unsigned char *output_asset, size_t output_asset_len, const unsigned char *output_abf, size_t output_abf_len, const unsigned char *output_generator, size_t output_generator_len, const unsigned char *bytes, size_t bytes_len, const unsigned char *asset, size_t asset_len, const unsigned char *abf, size_t abf_len, const unsigned char *generator, size_t generator_len, unsigned char *bytes_out, size_t len, size_t *written)

Generate an asset surjection proof.

Parameters

- **output_asset** – asset id for the output.
- **output_asset_len** – Length of asset. Must be ASSET_TAG_LEN.
- **output_abf** – Asset blinding factor for the output. Generated randomly for each output.
- **output_abf_len** – Length of output_abf. Must be BLINDING_FACTOR_LEN.
- **output_generator** – Asset generator from *wally_asset_generator_from_bytes*.
- **output_generator_len** – Length of output_generator`. Must be ``ASSET_GENERATOR_LEN.
- **bytes** – Must be generated randomly for each output.
- **bytes_len** – Length of bytes. Must be 32.
- **asset** – Array of input asset tags.
- **asset_len** – Length of asset`. Must be ``ASSET_TAG_LEN * number of inputs.
- **abf** – Array of input asset blinding factors.
- **abf_len** – Length of abf. Must be BLINDING_FACTOR_LEN * number of inputs.
- **generator** – Array of input asset generators.
- **generator_len** – Length of generator. Must be ASSET_GENERATOR_LEN * number of inputs.
- **bytes_out** – Buffer to receive surjection proof.
- **len** – Length of bytes_out. See *wally_asset_surjectionproof_len*.
- **written** – Number of bytes actually written to bytes_out.

Returns See *Variable Length Output Buffers*

int **wally_explicit_surjectionproof** (const unsigned char *output_asset, size_t output_asset_len, const unsigned char *output_abf, size_t output_abf_len, const unsigned char *output_generator, size_t output_generator_len, unsigned char *bytes_out, size_t len)

Generate an explicit asset surjection proof.

Parameters

- **output_asset** – asset id for the output.
- **output_asset_len** – Length of asset. Must be ASSET_TAG_LEN.
- **output_abf** – Asset blinding factor for the output. Generated randomly for each output.
- **output_abf_len** – Length of output_abf. Must be BLINDING_FACTOR_LEN.

- **output_generator** – Asset generator from `wally_asset_generator_from_bytes`.
- **output_generator_len** – Length of `output_generator`. Must be `ASSET_GENERATOR_LEN`.
- **bytes_out** – Buffer to receive surjection proof.
- **len** – Size of `bytes_out`. Must be `ASSET_EXPLICIT_SURJECTIONPROOF_LEN`.

Returns See *Error Codes*

`int wally_explicit_surjectionproof_verify` (const unsigned char *surjectionproof, size_t surjectionproof_len, const unsigned char *output_asset, size_t output_asset_len, const unsigned char *output_generator, size_t output_generator_len)

Verify an explicit asset surjection proof proves a given asset.

Parameters

- **surjectionproof** – The explicit asset surjection proof.
- **surjectionproof_len** – Length of `surjectionproof`.
- **output_asset** – The unblinded asset we expect `surjectionproof` to prove.
- **output_asset_len** – Length of `asset`. Must be `ASSET_TAG_LEN`.
- **output_generator** – Asset generator from `wally_asset_generator_from_bytes`.
- **output_generator_len** – Length of `output_generator`. Must be `ASSET_GENERATOR_LEN`.

Returns See *Error Codes*

`int wally_asset_unblind_with_nonce` (const unsigned char *nonce_hash, size_t nonce_hash_len, const unsigned char *proof, size_t proof_len, const unsigned char *commitment, size_t commitment_len, const unsigned char *extra, size_t extra_len, const unsigned char *generator, size_t generator_len, unsigned char *asset_out, size_t asset_out_len, unsigned char *abf_out, size_t abf_out_len, unsigned char *vbf_out, size_t vbf_out_len, uint64_t *value_out)

Unblind a confidential transaction output.

Parameters

- **nonce_hash** – SHA-256 hash of the generated nonce.
- **nonce_hash_len** – Length of `nonce_hash`. Must be `SHA256_LEN`.
- **proof** – Rangeproof from `wally_tx_get_output_rangeproof()`.
- **proof_len** – Length of `proof`.
- **commitment** – Value commitment from `wally_tx_get_output_value()`.
- **commitment_len** – Length of `commitment`.
- **extra** – Script pubkey from `wally_tx_get_output_script()`.
- **extra_len** – Length of `extra`.
- **generator** – Asset generator from `wally_tx_get_output_asset()`.
- **generator_len** – Length of `generator`. Must be `ASSET_GENERATOR_LEN`.

- **asset_out** – Buffer to receive unblinded asset id.
- **asset_out_len** – Size of **asset_out**. Must be `ASSET_TAG_LEN`.
- **abf_out** – Buffer to receive asset blinding factor.
- **abf_out_len** – Size of **abf_out**. Must be `BLINDING_FACTOR_LEN`.
- **vbf_out** – Buffer to receive asset blinding factor.
- **vbf_out_len** – Size of **vbf_out**. Must be `BLINDING_FACTOR_LEN`.
- **value_out** – Destination for unblinded transaction output value.

Returns See [Error Codes](#)

```
int wally_asset_unblind(const unsigned char *pub_key, size_t pub_key_len, const unsigned
char *priv_key, size_t priv_key_len, const unsigned char *proof,
size_t proof_len, const unsigned char *commitment, size_t commitment_len,
const unsigned char *extra, size_t extra_len, const unsigned char *generator,
size_t generator_len, unsigned char *asset_out, size_t asset_out_len, unsigned
char *abf_out, size_t abf_out_len, unsigned char *vbf_out, size_t vbf_out_len,
uint64_t *value_out)
```

Unblind a confidential transaction output.

Parameters

- **pub_key** – From `wally_tx_get_output_nonce()`.
- **pub_key_len** – Length of **pub_key**. Must be `EC_PUBLIC_KEY_LEN`.
- **priv_key** – Private blinding key corresponding to public blinding key used to generate destination address. See [wally_asset_blinding_key_to_ec_private_key\(\)](#).
- **proof** – Rangeproof from `wally_tx_get_output_rangeproof()`.
- **proof_len** – Length of **proof**.
- **commitment** – Value commitment from `wally_tx_get_output_value()`.
- **commitment_len** – Length of **commitment**.
- **extra** – Script pubkey from `wally_tx_get_output_script()`.
- **extra_len** – Length of **extra**.
- **generator** – Asset generator from `wally_tx_get_output_asset()`.
- **generator_len** – Length of **generator**. Must be `ASSET_GENERATOR_LEN`.
- **asset_out** – Buffer to receive unblinded asset id.
- **asset_out_len** – Size of **asset_out**. Must be `ASSET_TAG_LEN`.
- **abf_out** – Buffer to receive asset blinding factor.
- **abf_out_len** – Size of **abf_out**. Must be `BLINDING_FACTOR_LEN`.
- **vbf_out** – Buffer to receive asset blinding factor.
- **vbf_out_len** – Size of **vbf_out**. Must be `BLINDING_FACTOR_LEN`.
- **value_out** – Destination for unblinded transaction output value.

Returns See [Error Codes](#)

```
int wally_asset_blinding_key_from_seed(const unsigned char *bytes, size_t bytes_len, unsigned
char *bytes_out, size_t len)
```

Generate a master blinding key from a seed as specified in SLIP-0077.

Parameters

- **bytes** – Seed value. See `bip39_mnemonic_to_seed()`.
- **bytes_len** – Length of bytes. Must be one of BIP32_ENTROPY_LEN_128, BIP32_ENTROPY_LEN_256 or BIP32_ENTROPY_LEN_512.
- **bytes_out** – Buffer to receive master blinding key. The master blinding key can be used to generate blinding keys for specific outputs by passing it to `wally_asset_blinding_key_to_ec_private_key`.
- **len** – Size of bytes_out. Must be HMAC_SHA512_LEN.

Returns See *Error Codes*

```
int wally_asset_blinding_key_to_ec_private_key (const unsigned char *bytes,
                                              size_t bytes_len, const unsigned
                                              char *script, size_t script_len, unsigned
                                              char *bytes_out, size_t len)
```

Generate a blinding key for a script pubkey.

Parameters

- **bytes** – Master blinding key from `wally_asset_blinding_key_from_seed`.
- **bytes_len** – Length of bytes. Must be HMAC_SHA512_LEN.
- **script** – The script pubkey for the confidential output address.
- **script_len** – Length of script.
- **bytes_out** – Buffer to receive blinding key.
- **len** – Size of bytes_out. Must be EC_PRIVATE_KEY_LEN.

Returns See *Error Codes*

```
int wally_asset_pak_whitelistproof_size (size_t num_keys, size_t *written)
```

Calculate the size in bytes of a whitelist proof.

Parameters

- **num_keys** – The number of offline/online keys.
- **written** – Destination for the number of bytes needed for the proof.

Note: This function is a simpler variant of `wally_asset_pak_whitelistproof_len`.

Returns See *Error Codes*

```
int wally_asset_pak_whitelistproof (const unsigned char *online_keys, size_t online_keys_len,
                                   const unsigned char *offline_keys, size_t offline_keys_len,
                                   size_t key_index, const unsigned char *sub_pubkey,
                                   size_t sub_pubkey_len, const unsigned char *on-
                                   line_priv_key, size_t online_priv_key_len, const unsigned
                                   char *summed_key, size_t summed_key_len, unsigned
                                   char *bytes_out, size_t len, size_t *written)
```

Generate a whitelist proof for a pegout script.

Parameters

- **online_keys** – The list of concatenated online keys.

- **online_keys_len** – Length of `online_keys` in bytes. Must be a multiple of `EC_PUBLIC_KEY_LEN`.
- **offline_keys** – The list of concatenated offline keys.
- **offline_keys_len** – Length of `offline_keys` in bytes. Must match `online_keys_len`.
- **key_index** – The index in the PAK list of the key signing this whitelist proof.
- **sub_pubkey** – The public key to be whitelisted.
- **sub_pubkey_len** – Length of `sub_pubkey` in bytes. Must be `EC_PUBLIC_KEY_LEN`.
- **online_priv_key** – The secret key to the signer's online pubkey.
- **online_priv_key_len** – Length of `online_priv_key` in bytes. Must be `EC_PRIVATE_KEY_LEN`.
- **summed_key** – The secret key to the sum of (whitelisted key, signer's offline pubkey).
- **summed_key_len** – Length of `summed_key` in bytes. Must be `EC_PRIVATE_KEY_LEN`.
- **bytes_out** – Destination for the resulting whitelist proof.
- **len** – Length of `bytes_out` in bytes.
- **written** – Destination for the number of bytes written to `bytes_out`.

Returns See *Variable Length Output Buffers*

```
int wally_asset_pak_whitelistproof_len(const unsigned char *online_keys, size_t online_keys_len, const unsigned char *offline_keys, size_t offline_keys_len, size_t key_index, const unsigned char *sub_pubkey, size_t sub_pubkey_len, const unsigned char *online_priv_key, size_t online_priv_key_len, const unsigned char *summed_key, size_t summed_key_len, size_t *written)
```

Calculate the size in bytes of a whitelist proof.

Parameters

- **online_keys** – The list of concatenated online keys.
- **online_keys_len** – Length of `online_keys` in bytes. Must be a multiple of `EC_PUBLIC_KEY_LEN`.
- **offline_keys** – The list of concatenated offline keys.
- **offline_keys_len** – Length of `offline_keys` in bytes. Must match `online_keys_len`.
- **key_index** – The index in the PAK list of the key signing this whitelist proof.
- **sub_pubkey** – The public key to be whitelisted.
- **sub_pubkey_len** – Length of `sub_pubkey` in bytes. Must be `EC_PUBLIC_KEY_LEN`.
- **online_priv_key** – The secret key to the signer's online pubkey.
- **online_priv_key_len** – Length of `online_priv_key` in bytes. Must be `EC_PRIVATE_KEY_LEN`.
- **summed_key** – The secret key to the sum of (whitelisted key, signer's offline pubkey).

- **summed_key_len** – Length of summed_key in bytes. Must be EC_PRIVATE_KEY_LEN.
- **written** – Destination for resulting proof size in bytes.

Note: Use `wally_asset_pak_whitelistproof_size` for a simpler call interface.

Returns See *Error Codes*

Anti-Exfil Functions

```
int wally_ae_host_commit_from_bytes (const unsigned char *entropy, size_t entropy_len,
                                     uint32_t flags, unsigned char *bytes_out, size_t len)
```

Create the initial commitment to host randomness.

Parameters

- **entropy** – Randomness to commit to. It must come from a cryptographically secure RNG. As per the protocol, this value must not be revealed to the client until after the host has received the client commitment.
- **entropy_len** – The length of `entropy` in bytes. Must be `WALLY_S2C_DATA_LEN`.
- **flags** – Must be `EC_FLAG_ECDSA`.
- **bytes_out** – Destination for the resulting compact signature.
- **len** – Size of `bytes_out`. Must be `WALLY_HOST_COMMITMENT_LEN`.

Returns See [Error Codes](#)

```
int wally_ae_signer_commit_from_bytes (const unsigned char *priv_key, size_t priv_key_len,
                                       const unsigned char *bytes, size_t bytes_len, const
                                       unsigned char *commitment, size_t commitment_len,
                                       uint32_t flags, unsigned char *s2c_opening_out,
                                       size_t s2c_opening_out_len)
```

Compute signer's original nonce.

Parameters

- **priv_key** – The private key used for signing.
- **priv_key_len** – The length of `priv_key` in bytes. Must be `EC_PRIVATE_KEY_LEN`.
- **bytes** – The message hash to be signed.
- **bytes_len** – The length of `bytes` in bytes. Must be `EC_MESSAGE_HASH_LEN`.
- **commitment** – Randomness commitment from the host.

- **commitment_len** – The length of commitment in bytes. Must be `WALLY_HOST_COMMITMENT_LEN`.
- **flags** – Must be `EC_FLAG_ECDSA`.
- **s2c_opening_out** – Destination for the resulting opening information.
- **s2c_opening_out_len** – Size of `s2c_opening_out`. Must be `WALLY_S2C_OPENING_LEN`.

Returns See *Error Codes*

int **wally_ae_sig_from_bytes** (const unsigned char *priv_key, size_t priv_key_len, const unsigned char *bytes, size_t bytes_len, const unsigned char *entropy, size_t entropy_len, uint32_t flags, unsigned char *bytes_out, size_t len)
Same as `wally_ec_sig_from_bytes`, but commits to the host randomness.

Parameters

- **priv_key** – The private key to sign with.
- **priv_key_len** – The length of `priv_key` in bytes. Must be `EC_PRIVATE_KEY_LEN`.
- **bytes** – The message hash to sign.
- **bytes_len** – The length of `bytes` in bytes. Must be `EC_MESSAGE_HASH_LEN`.
- **entropy** – Host provided randomness.
- **entropy_len** – The length of `entropy` in bytes. Must be `WALLY_S2C_DATA_LEN`.
- **flags** – Must be `EC_FLAG_ECDSA`.
- **bytes_out** – Destination for the resulting compact signature.
- **len** – Size of `bytes_out`. Must be `EC_SIGNATURE_LEN`.

Returns See *Error Codes*

int **wally_ae_verify** (const unsigned char *pub_key, size_t pub_key_len, const unsigned char *bytes, size_t bytes_len, const unsigned char *entropy, size_t entropy_len, const unsigned char *s2c_opening, size_t s2c_opening_len, uint32_t flags, const unsigned char *sig, size_t sig_len)
Verify a signature was correctly constructed using the Anti-Exfil Protocol.

Parameters

- **pub_key** – The public key to verify with.
- **pub_key_len** – The length of `pub_key` in bytes. Must be `EC_PUBLIC_KEY_LEN`.
- **bytes** – The message hash to verify.
- **bytes_len** – The length of `bytes` in bytes. Must be `EC_MESSAGE_HASH_LEN`.
- **entropy** – Randomness provided by the host.
- **entropy_len** – The length of `entropy` in bytes. Must be `WALLY_S2C_DATA_LEN`.
- **s2c_opening** – Opening information provided by the signer.
- **s2c_opening_len** – The length of `s2c_opening` in bytes. Must be `WALLY_S2C_OPENING_LEN`.
- **flags** – Must be `EC_FLAG_ECDSA`.
- **sig** – The compact signature of the message in bytes.

- **sig_len** – The length of `sig` in bytes. Must be `EC_SIGNATURE_LEN`.

Returns See *Error Codes*

Library Conventions

15.1 Error Codes

The following values can be returned by library functions:

Code	Meaning
WALLY_OK	The function completed without error. See <i>Variable Length Output Buffers</i> if applicable for the function.
WALLY_ERROR	An internal or unexpected error happened in the library. In some cases this code may indicate a specific error condition which will be documented with the function.
WALLY_EINVAL	One or more parameters passed to the function is not valid. For example, a required buffer value is NULL or of the wrong length.
WALLY_ENOMEM	The function required memory allocation but no memory could be allocated from the O/S.

15.2 Variable Length Output Buffers

Some functions write output that can vary in length to user supplied buffers. In these cases, the number of written bytes is placed in the `written` output parameter when the function completes.

If the user supplied buffer is of insufficient size, these functions will still return `WALLY_OK`, but will place the required size in the `written` output parameter.

Callers must check not only that the function succeeds, but also that the number of bytes written is less than or equal to the supplied buffer size. If the buffer is too small, it should be resized to the returned size and the call retried.

The following walkthrough demonstrates how to use libwally to create a transaction spending a confidential liquid utxo. For documentation of the Blockstream Liquid network please go to [Blockstream](#)

The example code here is written in python using the generated python swig wrappers.

16.1 Generating a confidential address

Start by creating a standard p2pkh address. Assume that we have defined `mnemonic` as a 24 word mnemonic for the wallet we want to use. From this we can derive bip32 keys depending on the requirements of the wallet.

```
seed = wally.bip39_mnemonic_to_seed512(mnemonic, '')
wallet_master_key = wally.bip32_key_from_seed(
    seed,
    wally.BIP32_VER_TEST_PRIVATE, 0)
wallet_derived_key = wally.bip32_key_from_parent(
    wallet_master_key,
    1,
    wally.BIP32_FLAG_KEY_PRIVATE)
address = wally.bip32_key_to_address(
    wallet_derived_key,
    wally.WALLY_ADDRESS_TYPE_P2PKH,
    wally.WALLY_ADDRESS_VERSION_P2PKH_LIQUID_REGTEST)
```

For each new receive address a blinding key should be deterministically derived from a master blinding key, itself derived from the bip39 mnemonic for the wallet. wally provides the function `wally.asset_blinding_key_from_seed()` which can be used to derive a master blinding key from a mnemonic.

```
master_blinding_key = wally.asset_blinding_key_from_seed(seed)
script_pubkey = wally.address_to_scriptpubkey(
    address,
    wally.WALLY_NETWORK_LIQUID_REGTEST)
```

(continues on next page)

(continued from previous page)

```
private_blinding_key = wally.asset_blinding_key_to_ec_private_key(
    master_blinding_key,
    script_pubkey)
public_blinding_key = wally.ec_public_key_from_private_key(
    private_blinding_key)
```

Finally call the wally function `wally.confidential_addr_from_addr()` to combine the non-confidential address with the public blinding key to create a confidential address. We also supply a blinding prefix indicating the network version.

```
confidential_address = wally.confidential_addr_from_addr(
    address,
    wally.WALLY_CA_PREFIX_LIQUID_REGTEST,
    public_blinding_key)
```

The confidential address can now be passed to liquid-cli to receive funds. We'll send 1.1 BTC to our confidential address and save the raw hex transaction for further processing.

```
$ liquid-cli getrawtransaction $(sendtoaddress <confidential address> 1.1)
```

16.2 Receiving confidential assets

On receiving confidential (blinded) utxos you can use libwally to unblind and inspect them. Take the hex transaction returned by getrawtransaction above and create a libwally tx.

```
tx = wally.tx_from_hex(
    tx_hex,
    wally.WALLY_TX_FLAG_USE_ELEMENTS | wally.WALLY_TX_FLAG_USE_WITNESS)
```

The transaction will likely have three outputs: the utxo paying to our confidential address, a change output and an explicit fee output(Liquid transactions differ from standard bitcoin transaction in that the fee is an explicit output). Iterate over the transaction outputs and unblind any addressed to us.

```
asset_generators_in = b''
asset_ids_in = b''
values_in = []
abfs_in = b''
vbfs_in = b''
script_pubkeys_in = []
vouts_in = []
num_outputs = wally.tx_get_num_outputs(tx)
for vout in range(num_outputs):
    script_pubkey_out = wally.tx_get_output_script(tx, vout)
    if script_pubkey_out != script_pubkey:
        continue

    vouts_in.append(vout)

    sender_ephemeral_pubkey = wally.tx_get_output_nonce(tx, vout)
    rangeproof = wally.tx_get_output_rangeproof(tx, vout)
    script_pubkey = wally.tx_get_output_script(tx, vout)
    asset_id = wally.tx_get_output_asset(tx, vout)
    value_commitment = wally.tx_get_output_value(tx, vout)
```

(continues on next page)

(continued from previous page)

```

script_pubkeys_in.append(script_pubkey)

value, asset_id, abf, vbf = wally.asset_unblind(
    sender_ephemeral_pubkey,
    private_blinding_key,
    rangeproof,
    value_commitment,
    script_pubkey,
    asset_id)

asset_generator = wally.asset_generator_from_bytes(asset_id, abf)

asset_generators_in += asset_generator
asset_ids_in += asset_id
values_in.append(value)
abfs_in += abf
vbfs_in += vbf

```

We have now unblinded the values and asset ids of the utxos. We've also saved the abfs (asset blinding factors) and vbfs (value binding factors) because they are needed for the next step: spending the utxos.

16.3 Spending confidential assets

The wallet logic will define the transaction outputs, values and fees. Here we assume that we're only dealing with a single asset and a single confidential recipient address `destination_address` to which we'll pay the input amount less some fixed fee.

```

total_in = sum(values_in)
output_values = [total_in - fee,]
confidential_output_addresses = [destination_address,]
output_asset_ids = asset_ids_in[:wally.ASSET_TAG_LEN]

```

Generate blinding factors for the outputs. These are asset blinding factors (abf) and value blinding factors (vbf). The blinding factors are random except for the final vbf which must be calculated by calling `wally.asset_final_vbf()`.

```

num_inputs = len(values_in)
num_outputs = 1
abfs_out = os.urandom(32 * num_outputs)
vbfs_out = os.urandom(32 * (num_outputs - 1))
vbfs_out += wally.asset_final_vbf(
    values_in + output_values,
    num_inputs,
    abfs_in + abfs_out,
    vbfs_in + vbfs_out)

```

A confidential output address can be decomposed into a standard address plus the public blinding key, and the libwally function `wally.address_to_scriptpubkey()` will provide the corresponding script pubkey.

```

blinding_pubkeys = [
    wally.confidential_addr_to_ec_public_key(
        confidential_address, address_prefix)
    for confidential_address in confidential_output_addresses]

```

(continues on next page)

(continued from previous page)

```

non_confidential_addresses = [
    wally.confidential_addr_to_addr(
        confidential_address, address_prefix)
    for confidential_address in confidential_output_addresses]

script_pubkeys = [
    wally.address_to_scriptpubkey(
        non_confidential_address, network)
    for non_confidential_address in non_confidential_addresses]

```

Create a new transaction

```

version = 2
locktime = 0
output_tx = wally.tx_init(version, locktime, 0, 0)

```

Iterate over the outputs and calculate the value commitment, rangeproof and surjectionproofs. This requires generating a random ephemeral public/private ec key pair for each blinded output.

```

for value, blinding_pubkey, script_pubkey in zip(
    output_values, blinding_pubkeys, script_pubkeys):

    abf, abfs_out = abfs_out[:32], abfs_out[32:]
    vbf, vbfs_out = vbfs_out[:32], vbfs_out[32:]
    asset_id, output_asset_ids = output_asset_ids[:32], output_asset_ids[32:]

    generator = wally.asset_generator_from_bytes(asset_id, abf)

    value_commitment = wally.asset_value_commitment(
        value, vbf, generator)

    ephemeral_privkey = os.urandom(32)
    ephemeral_pubkey = wally.ec_public_key_from_private_key(
        ephemeral_privkey)

    rangeproof = wally.asset_rangeproof(
        value,
        blinding_pubkey,
        ephemeral_privkey,
        asset_id,
        abf,
        vbf,
        value_commitment,
        script_pubkey,
        generator,
        1, # min_value
        0, # exponent
        36 # bits
    )

    surjectionproof = wally.asset_surjectionproof(
        asset_id,
        abf,
        generator,
        os.urandom(32),

```

(continues on next page)

(continued from previous page)

```

        asset_ids_in,
        abfs_in,
        asset_generators_in
    )

    wally.tx_add_elements_raw_output(
        output_tx,
        script_pubkey,
        generator,
        value_commitment,
        ephemeral_pubkey,
        surjectionproof,
        rangeproof,
        0
    )

```

Finally the fee output must be explicitly added (unlike standard Bitcoin transactions where the fee is implicit). The fee is always payable in the bitcoin asset. The wallet logic will determine the fee amount.

```

BITCOIN = "5ac9f65c0efcc4775e0baec4ec03abdde22473cd3cf33c0419ca290e0751b225"
BITCOIN = wally.hex_to_bytes(BITCOIN)[: -1]

wally.tx_add_elements_raw_output(
    output_tx,
    None,
    bytearray([0x01]) + BITCOIN,
    wally.tx_confidential_value_from_satoshi(fee),
    None, # nonce
    None, # surjection proof
    None, # range proof
    0)

```

Sign the transaction inputs.

```

flags = 0
prev_txid = wally.tx_get_txid(tx)
for vout in vouts_in:

    wally.tx_add_elements_raw_input(
        output_tx,
        prev_txid,
        vout,
        0xffffffff,
        None, # scriptSig
        None, # witness
        None, # nonce
        None, # entropy
        None, # issuance amount
        None, # inflation keys
        None, # issuance amount rangeproof
        None, # inflation keys rangeproof
        None, # pegin witness
        flags)

for vin, script_pubkey in enumerate(script_pubkeys_in):

```

(continues on next page)

(continued from previous page)

```
privkey = wally.bip32_key_get_priv_key(wallet_derived_key)
pubkey = wally.ec_public_key_from_private_key(privkey)

sighash = wally.tx_get_elements_signature_hash(
    output_tx, vin, script_pubkey, None, wally.WALLY_SIGHASH_ALL, 0)

signature = wally.ec_sig_from_bytes(
    privkey, sighash, wally.EC_FLAG_ECDSA)

scriptsig = wally.scriptsig_p2pkh_from_sig(
    pubkey, signature, wally.WALLY_SIGHASH_ALL)

wally.tx_set_input_script(output_tx, vin, scriptsig)
```

The transaction is now ready to be broadcast, the hex value is easily retrieved by calling `wally.tx_to_hex()`.

```
tx_hex = wally.tx_to_hex(output_tx, wally.WALLY_TX_FLAG_USE_WITNESS)
```

CHAPTER 17

Anti-Exfil Protocol

The following walkthrough demonstrates how to use libwally to implement the ECDSA Anti-Exfil Protocol to prevent a signing device from exfiltrating the secret signing keys through biased signature nonces. For the full details, see [here](#).

The example code here is written in python using the generated python swig wrappers.

17.1 Step 1

The host draws randomness `rho` and computes a commitment to it:

```
host_commitment = wally.ae_host_commit_from_bytes(rho, wally.EC_FLAG_ECDSA)
```

Host sends `host_commitment` to the signer.

17.2 Step 2

The signing device computes the original nonce `R`, i.e. `signer_commitment`:

```
signer_commitment = wally.ae_signer_commit_from_bytes(priv_key, message_hash, host_
→commitment, wally.EC_FLAG_ECDSA)
```

Signing device sends `signer_commitment` to the host.

Warning: If, at any point from this step onward, the hardware device fails, it is okay to restart the protocol using **exactly the same** `rho` and checking that the hardware device proposes **exactly the same** `R`. Otherwise, the hardware device may be selectively aborting and thereby biasing the set of nonces that are used in actual signatures.

It takes many (>100) such aborts before there is a plausible attack, given current knowledge in 2020. However such aborts accumulate even across a total replacement of all relevant devices (but not across replacement of the actual signing keys with new independently random ones).

In case the hardware device cannot be made to sign with the given `rho`, `R` pair, wallet authors should alert the user and present a very scary message implying that if this happens more than even a few times, say 20 or more times EVER, they should change hardware vendors and perhaps sweep their coins.

17.3 Step 3

The host replies with `rho` generated at *Step 1*.

17.4 Step 4

The signing device signs and provide the signature to the host:

```
signature = wally.ae_sig_from_bytes(priv_key, message_hash, rho, wally.EC_FLAG_ECDSA)
```

17.5 Step 5

The host verifies that the signature's public nonce matches the signer commitment `R` from *Step 2* and its original randomness `rho`:

```
wally.ae_verify(pub_key, message_hash, rho, signer_commitment, wally.EC_FLAG_ECDSA,   
↪signature)
```


CHAPTER 18

Indices and tables

- `genindex`
- `search`

B

bip32_key_free (*C function*), 25
 bip32_key_from_base58 (*C function*), 29
 bip32_key_from_base58_alloc (*C function*), 29
 bip32_key_from_base58_n (*C function*), 29
 bip32_key_from_base58_n_alloc (*C function*), 29
 bip32_key_from_parent (*C function*), 27
 bip32_key_from_parent_alloc (*C function*), 27
 bip32_key_from_parent_path (*C function*), 27
 bip32_key_from_parent_path_alloc (*C function*), 27
 bip32_key_from_parent_path_str (*C function*), 28
 bip32_key_from_parent_path_str_alloc (*C function*), 28
 bip32_key_from_parent_path_str_n (*C function*), 28
 bip32_key_from_parent_path_str_n_alloc (*C function*), 28
 bip32_key_from_seed (*C function*), 26
 bip32_key_from_seed_alloc (*C function*), 26
 bip32_key_from_seed_custom (*C function*), 25
 bip32_key_from_seed_custom_alloc (*C function*), 26
 bip32_key_get_fingerprint (*C function*), 30
 bip32_key_init (*C function*), 25
 bip32_key_init_alloc (*C function*), 25
 bip32_key_serialize (*C function*), 26
 bip32_key_strip_private_key (*C function*), 30
 bip32_key_to_base58 (*C function*), 29
 bip32_key_unserialize (*C function*), 26
 bip32_key_unserialize_alloc (*C function*), 27
 bip32_key_with_tweak_from_parent_path (*C function*), 28
 bip32_key_with_tweak_from_parent_path_alloc (*C function*), 29
 bip38_from_private_key (*C function*), 31
 bip38_get_flags (*C function*), 32

bip38_raw_from_private_key (*C function*), 31
 bip38_raw_get_flags (*C function*), 32
 bip38_raw_to_private_key (*C function*), 32
 bip38_to_private_key (*C function*), 32
 bip39_get_languages (*C function*), 33
 bip39_get_word (*C function*), 33
 bip39_get_wordlist (*C function*), 33
 bip39_mnemonic_from_bytes (*C function*), 33
 bip39_mnemonic_to_bytes (*C function*), 34
 bip39_mnemonic_to_seed (*C function*), 34
 bip39_mnemonic_to_seed512 (*C function*), 34
 bip39_mnemonic_validate (*C function*), 34

W

wally_addr_segwit_from_bytes (*C function*), 19
 wally_addr_segwit_get_version (*C function*), 20
 wally_addr_segwit_n_get_version (*C function*), 20
 wally_addr_segwit_n_to_bytes (*C function*), 19
 wally_addr_segwit_to_bytes (*C function*), 19
 wally_address_to_scriptpubkey (*C function*), 20
 wally_ae_host_commit_from_bytes (*C function*), 143
 wally_ae_sig_from_bytes (*C function*), 144
 wally_ae_signer_commit_from_bytes (*C function*), 143
 wally_ae_verify (*C function*), 144
 wally_aes (*C function*), 7
 wally_aes_cbc (*C function*), 8
 wally_asset_blinding_key_from_seed (*C function*), 138
 wally_asset_blinding_key_to_ec_private_key (*C function*), 139
 wally_asset_final_vbf (*C function*), 132
 wally_asset_generator_from_bytes (*C function*), 131

wally_asset_pak_whitelistproof (C function), 139

wally_asset_pak_whitelistproof_len (C function), 140

wally_asset_pak_whitelistproof_size (C function), 139

wally_asset_rangeproof (C function), 134

wally_asset_rangeproof_with_nonce (C function), 133

wally_asset_scalar_offset (C function), 132

wally_asset_surjectionproof (C function), 136

wally_asset_surjectionproof_len (C function), 135

wally_asset_surjectionproof_size (C function), 135

wally_asset_unblind (C function), 138

wally_asset_unblind_with_nonce (C function), 137

wally_asset_value_commitment (C function), 133

wally_base58_from_bytes (C function), 3

wally_base58_get_length (C function), 4

wally_base58_n_get_length (C function), 4

wally_base58_n_to_bytes (C function), 3

wally_base58_to_bytes (C function), 3

wally_base64_from_bytes (C function), 4

wally_base64_get_maximum_length (C function), 4

wally_base64_to_bytes (C function), 4

wally_bip32_key_to_addr_segwit (C function), 22

wally_bip32_key_to_address (C function), 22

wally_bip340_tagged_hash (C function), 9

wally_bzero (C function), 1

wally_cleanup (C function), 1

wally_confidential_addr_from_addr (C function), 23

wally_confidential_addr_from_addr_segwit (C function), 24

wally_confidential_addr_segwit_to_ec_public_key (C function), 23

wally_confidential_addr_to_addr (C function), 22

wally_confidential_addr_to_addr_segwit (C function), 23

wally_confidential_addr_to_ec_public_key (C function), 23

wally_descriptor_canonicalize (C function), 101

wally_descriptor_free (C function), 101

wally_descriptor_get_checksum (C function), 102

wally_descriptor_get_features (C function), 102

wally_descriptor_get_network (C function), 102

wally_descriptor_get_num_paths (C function), 103

wally_descriptor_get_num_variants (C function), 103

wally_descriptor_parse (C function), 101

wally_descriptor_set_network (C function), 102

wally_descriptor_to_address (C function), 104

wally_descriptor_to_addresses (C function), 105

wally_descriptor_to_script (C function), 104

wally_descriptor_to_script_get_maximum_length (C function), 103

wally_ec_private_key_verify (C function), 11

wally_ec_public_key_decompress (C function), 12

wally_ec_public_key_from_private_key (C function), 11

wally_ec_public_key_negate (C function), 12

wally_ec_public_key_verify (C function), 11

wally_ec_scalar_add (C function), 14

wally_ec_scalar_add_to (C function), 16

wally_ec_scalar_multiply (C function), 15

wally_ec_scalar_multiply_by (C function), 16

wally_ec_scalar_subtract (C function), 15

wally_ec_scalar_subtract_from (C function), 16

wally_ec_scalar_verify (C function), 14

wally_ec_sig_from_bytes (C function), 13

wally_ec_sig_from_bytes_len (C function), 12

wally_ec_sig_from_der (C function), 13

wally_ec_sig_normalize (C function), 13

wally_ec_sig_to_der (C function), 13

wally_ec_sig_to_public_key (C function), 14

wally_ec_sig_verify (C function), 14

wally_ec_xonly_public_key_verify (C function), 11

wally_ecdh (C function), 17

wally_ecdh_nonce_hash (C function), 131

wally_elements_pegin_contract_script_from_bytes (C function), 99

wally_elements_pegout_script_from_bytes (C function), 98

wally_elements_pegout_script_size (C function), 98

wally_explicit_rangeproof (C function), 134

wally_explicit_rangeproof_verify (C function), 134

wally_explicit_surjectionproof (C function), 136

[wally_explicit_surjectionproof_verify \(C function\), 137](#)
[wally_format_bitcoin_message \(C function\), 16](#)
[wally_free_string \(C function\), 2](#)
[wally_get_new_secp_context \(C function\), 1](#)
[wally_get_operations \(C function\), 5](#)
[wally_get_secp_context \(C function\), 1](#)
[wally_hash160 \(C function\), 9](#)
[wally_hex_from_bytes \(C function\), 2](#)
[wally_hex_n_to_bytes \(C function\), 3](#)
[wally_hex_n_verify \(C function\), 2](#)
[wally_hex_to_bytes \(C function\), 2](#)
[wally_hex_verify \(C function\), 2](#)
[wally_hmac_sha256 \(C function\), 10](#)
[wally_hmac_sha512 \(C function\), 10](#)
[wally_init \(C function\), 1](#)
[wally_is_elements_build \(C function\), 5](#)
[wally_keypath_bip32_verify \(C function\), 42](#)
[wally_keypath_get_fingerprint \(C function\), 44](#)
[wally_keypath_get_path \(C function\), 45](#)
[wally_keypath_get_path_len \(C function\), 44](#)
[wally_keypath_public_key_verify \(C function\), 43](#)
[wally_keypath_xonly_public_key_verify \(C function\), 43](#)
[wally_map_add \(C function\), 38](#)
[wally_map_add_integer \(C function\), 38](#)
[wally_map_assign \(C function\), 41](#)
[wally_map_clear \(C function\), 37](#)
[wally_map_combine \(C function\), 41](#)
[wally_map_find \(C function\), 39](#)
[wally_map_find_bip32_public_key_from \(C function\), 42](#)
[wally_map_find_from \(C function\), 39](#)
[wally_map_find_integer \(C function\), 39](#)
[wally_map_free \(C function\), 37](#)
[wally_map_get \(C function\), 39](#)
[wally_map_get_integer \(C function\), 39](#)
[wally_map_get_item \(C function\), 41](#)
[wally_map_get_item_integer_key \(C function\), 40](#)
[wally_map_get_item_key \(C function\), 40](#)
[wally_map_get_item_key_length \(C function\), 40](#)
[wally_map_get_item_length \(C function\), 41](#)
[wally_map_get_num_items \(C function\), 40](#)
[wally_map_hash_preimage_verify \(C function\), 46](#)
[wally_map_init \(C function\), 37](#)
[wally_map_init_alloc \(C function\), 37](#)
[wally_map_keypath_add \(C function\), 43](#)
[wally_map_keypath_bip32_init_alloc \(C function\), 43](#)
[wally_map_keypath_get_bip32_key_from_alloc \(C function\), 42](#)
[wally_map_keypath_get_item_fingerprint \(C function\), 44](#)
[wally_map_keypath_get_item_path \(C function\), 45](#)
[wally_map_keypath_get_item_path_len \(C function\), 45](#)
[wally_map_keypath_public_key_init_alloc \(C function\), 43](#)
[wally_map_preimage_hash160_add \(C function\), 46](#)
[wally_map_preimage_init_alloc \(C function\), 46](#)
[wally_map_preimage_ripemd160_add \(C function\), 46](#)
[wally_map_preimage_sha256_add \(C function\), 46](#)
[wally_map_preimage_sha256d_add \(C function\), 47](#)
[wally_map_remove \(C function\), 38](#)
[wally_map_remove_integer \(C function\), 39](#)
[wally_map_replace \(C function\), 38](#)
[wally_map_replace_integer \(C function\), 38](#)
[wally_map_sort \(C function\), 41](#)
[wally_pbkdf2_hmac_sha256 \(C function\), 10](#)
[wally_pbkdf2_hmac_sha512 \(C function\), 11](#)
[wally_psbt_add_global_scalar \(C function\), 82](#)
[wally_psbt_add_tx_input_at \(C function\), 83](#)
[wally_psbt_add_tx_output_at \(C function\), 85](#)
[wally_psbt_blind \(C function\), 87](#)
[wally_psbt_blind_alloc \(C function\), 87](#)
[wally_psbt_clear_fallback_locktime \(C function\), 82](#)
[wally_psbt_clone_alloc \(C function\), 87](#)
[wally_psbt_combine \(C function\), 86](#)
[wally_psbt_extract \(C function\), 89](#)
[wally_psbt_finalize \(C function\), 88](#)
[wally_psbt_find_global_scalar \(C function\), 82](#)
[wally_psbt_free \(C function\), 80](#)
[wally_psbt_from_base64 \(C function\), 86](#)
[wally_psbt_from_bytes \(C function\), 85](#)
[wally_psbt_get_id \(C function\), 80](#)
[wally_psbt_get_input_bip32_key_from_alloc \(C function\), 83](#)
[wally_psbt_get_input_scriptcode \(C function\), 84](#)
[wally_psbt_get_input_scriptcode_len \(C function\), 84](#)
[wally_psbt_get_input_signature_hash \(C](#)

function), 85

wally_psbt_get_input_signing_script (*C function*), 84

wally_psbt_get_input_signing_script_len (*C function*), 84

wally_psbt_get_length (*C function*), 86

wally_psbt_get_locktime (*C function*), 81

wally_psbt_get_tx_version (*C function*), 81

wally_psbt_init_alloc (*C function*), 79

wally_psbt_input_add_signature (*C function*), 52

wally_psbt_input_clear_amount_rangeproof (*C function*), 55

wally_psbt_input_clear_asset (*C function*), 55

wally_psbt_input_clear_asset_surjectionproof (*C function*), 56

wally_psbt_input_clear_inflation_keys_blinding (*C function*), 67

wally_psbt_input_clear_inflation_keys_commitment (*C function*), 65

wally_psbt_input_clear_inflation_keys_rangeproof (*C function*), 66

wally_psbt_input_clear_issuance_amount_blinding (*C function*), 64

wally_psbt_input_clear_issuance_amount_commitment (*C function*), 61

wally_psbt_input_clear_issuance_amount_rangeproof (*C function*), 61

wally_psbt_input_clear_issuance_asset_entropy (*C function*), 63

wally_psbt_input_clear_issuance_blinding_nonce (*C function*), 62

wally_psbt_input_clear_pegin_claim_script (*C function*), 60

wally_psbt_input_clear_pegin_genesis_blockhash (*C function*), 59

wally_psbt_input_clear_pegin_txout_proof (*C function*), 58

wally_psbt_input_clear_required_lockheight (*C function*), 53

wally_psbt_input_clear_required_locktime (*C function*), 53

wally_psbt_input_clear_sequence (*C function*), 49

wally_psbt_input_clear_utxo_rangeproof (*C function*), 68

wally_psbt_input_find_keypath (*C function*), 51

wally_psbt_input_find_signature (*C function*), 52

wally_psbt_input_find_unknown (*C function*), 53

wally_psbt_input_generate_explicit_proofs (*C function*), 68

wally_psbt_input_get_amount_rangeproof (*C function*), 54

wally_psbt_input_get_amount_rangeproof_len (*C function*), 54

wally_psbt_input_get_asset (*C function*), 55

wally_psbt_input_get_asset_len (*C function*), 55

wally_psbt_input_get_asset_surjectionproof (*C function*), 56

wally_psbt_input_get_asset_surjectionproof_len (*C function*), 56

wally_psbt_input_get_inflation_keys_blinding_rangeproof (*C function*), 66

wally_psbt_input_get_inflation_keys_blinding_rangeproof_len (*C function*), 67

wally_psbt_input_get_inflation_keys_commitment (*C function*), 65

wally_psbt_input_get_inflation_keys_commitment_len (*C function*), 65

wally_psbt_input_get_inflation_keys_rangeproof (*C function*), 66

wally_psbt_input_get_inflation_keys_rangeproof_len (*C function*), 66

wally_psbt_input_get_issuance_amount_blinding_rangeproof (*C function*), 63

wally_psbt_input_get_issuance_amount_blinding_rangeproof_len (*C function*), 64

wally_psbt_input_get_issuance_amount_commitment (*C function*), 60

wally_psbt_input_get_issuance_amount_commitment_len (*C function*), 60

wally_psbt_input_get_issuance_amount_rangeproof (*C function*), 61

wally_psbt_input_get_issuance_amount_rangeproof_len (*C function*), 61

wally_psbt_input_get_issuance_asset_entropy (*C function*), 62

wally_psbt_input_get_issuance_asset_entropy_len (*C function*), 63

wally_psbt_input_get_issuance_blinding_nonce (*C function*), 62

wally_psbt_input_get_issuance_blinding_nonce_len (*C function*), 62

wally_psbt_input_get_pegin_claim_script (*C function*), 59

wally_psbt_input_get_pegin_claim_script_len (*C function*), 59

wally_psbt_input_get_pegin_genesis_blockhash (*C function*), 58

wally_psbt_input_get_pegin_genesis_blockhash_len (*C function*), 58

wally_psbt_input_get_pegin_txout_proof (*C function*), 57

wally_psbt_input_get_pegin_txout_proof_len (C function), 58	wally_psbt_input_set_pegin_witness (C function), 57
wally_psbt_input_get_utxo_rangeproof (C function), 67	wally_psbt_input_set_previous_txid (C function), 49
wally_psbt_input_get_utxo_rangeproof_len (C function), 68	wally_psbt_input_set_redeem_script (C function), 50
wally_psbt_input_is_finalized (C function), 69	wally_psbt_input_set_required_lockheight (C function), 53
wally_psbt_input_keypath_add (C function), 51	wally_psbt_input_set_required_locktime (C function), 53
wally_psbt_input_set_amount (C function), 54	wally_psbt_input_set_sequence (C function), 49
wally_psbt_input_set_amount_rangeproof (C function), 54	wally_psbt_input_set_sighash (C function), 53
wally_psbt_input_set_asset (C function), 55	wally_psbt_input_set_signatures (C function), 52
wally_psbt_input_set_asset_surjectionproof (C function), 56	wally_psbt_input_set_unknowns (C function), 52
wally_psbt_input_set_final_scriptsig (C function), 50	wally_psbt_input_set_utxo (C function), 50
wally_psbt_input_set_final_witness (C function), 51	wally_psbt_input_set_utxo_rangeproof (C function), 68
wally_psbt_input_set_inflation_keys (C function), 57	wally_psbt_input_set_witness_script (C function), 50
wally_psbt_input_set_inflation_keys_blinding_rangeproof (C function), 67	wally_psbt_input_set_witness_utxo (C function), 50
wally_psbt_input_set_inflation_keys_commitment (C function), 65	wally_psbt_input_set_witness_utxo_from_tx (C function), 50
wally_psbt_input_set_inflation_keys_rangeproof (C function), 66	wally_psbt_is_elements (C function), 89
wally_psbt_input_set_issuance_amount (C function), 56	wally_psbt_is_finalized (C function), 81
wally_psbt_input_set_issuance_amount_blinding_rangeproof (C function), 64	wally_psbt_output_clear_amount (C function), 71
wally_psbt_input_set_issuance_amount_commitment (C function), 60	wally_psbt_output_clear_asset (C function), 73
wally_psbt_input_set_issuance_amount_rangeproof (C function), 61	wally_psbt_output_clear_asset_blinding_surjectionproof (C function), 79
wally_psbt_input_set_issuance_asset_entropy (C function), 63	wally_psbt_output_clear_asset_commitment (C function), 74
wally_psbt_input_set_issuance_blinding_nonce (C function), 62	wally_psbt_output_clear_asset_surjectionproof (C function), 75
wally_psbt_input_set_keypaths (C function), 51	wally_psbt_output_clear_blinder_index (C function), 71
wally_psbt_input_set_output_index (C function), 49	wally_psbt_output_clear_blinding_public_key (C function), 76
wally_psbt_input_set_pegin_amount (C function), 57	wally_psbt_output_clear_ecdh_public_key (C function), 77
wally_psbt_input_set_pegin_claim_script (C function), 59	wally_psbt_output_clear_value_blinding_rangeproof (C function), 78
wally_psbt_input_set_pegin_genesis_blockhash (C function), 59	wally_psbt_output_clear_value_commitment (C function), 72
wally_psbt_input_set_pegin_tx (C function), 57	wally_psbt_output_clear_value_rangeproof (C function), 74
wally_psbt_input_set_pegin_txout_proof (C function), 58	wally_psbt_output_find_keypath (C function), 69
	wally_psbt_output_find_unknown (C function), 70

tion), 70

wally_psbt_output_get_asset (*C function*), 72

wally_psbt_output_get_asset_blinding_surjectionproof (*C function*), 78

wally_psbt_output_get_asset_blinding_surjectionproof_len (*C function*), 78

wally_psbt_output_get_asset_commitment (*C function*), 73

wally_psbt_output_get_asset_commitment_len (*C function*), 73

wally_psbt_output_get_asset_len (*C function*), 72

wally_psbt_output_get_asset_surjectionproof (*C function*), 75

wally_psbt_output_get_asset_surjectionproof_len (*C function*), 75

wally_psbt_output_get_blinding_public_key (*C function*), 75

wally_psbt_output_get_blinding_public_key_len (*C function*), 76

wally_psbt_output_get_blinding_status (*C function*), 79

wally_psbt_output_get_ecdh_public_key (*C function*), 76

wally_psbt_output_get_ecdh_public_key_len (*C function*), 77

wally_psbt_output_get_value_blinding_rangeproof (*C function*), 77

wally_psbt_output_get_value_blinding_rangeproof_len (*C function*), 78

wally_psbt_output_get_value_commitment (*C function*), 71

wally_psbt_output_get_value_commitment_len (*C function*), 72

wally_psbt_output_get_value_rangeproof (*C function*), 74

wally_psbt_output_get_value_rangeproof_len (*C function*), 74

wally_psbt_output_keypath_add (*C function*), 70

wally_psbt_output_set_amount (*C function*), 70

wally_psbt_output_set_asset (*C function*), 73

wally_psbt_output_set_asset_blinding_surjectionproof (*C function*), 79

wally_psbt_output_set_asset_commitment (*C function*), 73

wally_psbt_output_set_asset_surjectionproof (*C function*), 75

wally_psbt_output_set_blinder_index (*C function*), 71

wally_psbt_output_set_blinding_public_key (*C function*), 76

wally_psbt_output_set_ecdh_public_key (*C function*), 77

wally_psbt_output_set_keypaths (*C function*), 69

wally_psbt_output_set_redeem_script (*C function*), 69

wally_psbt_output_set_script (*C function*), 71

wally_psbt_output_set_unknowns (*C function*), 70

wally_psbt_output_set_value_blinding_rangeproof (*C function*), 78

wally_psbt_output_set_value_commitment (*C function*), 72

wally_psbt_output_set_value_rangeproof (*C function*), 74

wally_psbt_output_set_witness_script (*C function*), 69

wally_psbt_remove_input (*C function*), 83

wally_psbt_remove_output (*C function*), 85

wally_psbt_set_fallback_locktime (*C function*), 82

wally_psbt_set_global_scalars (*C function*), 82

wally_psbt_set_global_tx (*C function*), 81

wally_psbt_set_pset_modifiable_flags (*C function*), 83

wally_psbt_set_tx_modifiable_flags (*C function*), 82

wally_psbt_set_tx_version (*C function*), 81

wally_psbt_set_version (*C function*), 80

wally_psbt_sign (*C function*), 87

wally_psbt_sign_bip32 (*C function*), 88

wally_psbt_sign_input_bip32 (*C function*), 88

wally_psbt_to_base64 (*C function*), 86

wally_psbt_to_bytes (*C function*), 86

wally_ripemd160 (*C function*), 9

wally_s2c_commitment_verify (*C function*), 17

wally_s2c_sig_from_bytes (*C function*), 17

wally_script_push_from_bytes (*C function*), 96

wally_scriptpubkey_csv_2of2_then_1_from_bytes (*C function*), 95

wally_scriptpubkey_csv_2of2_then_1_from_bytes_opt (*C function*), 95

wally_scriptpubkey_csv_2of3_then_2_from_bytes (*C function*), 96

wally_scriptpubkey_get_type (*C function*), 91

wally_scriptpubkey_multisig_from_bytes (*C function*), 94

wally_scriptpubkey_op_return_from_bytes (*C function*), 93

wally_scriptpubkey_p2pkh_from_bytes (*C function*), 91

wally_scriptpubkey_p2sh_from_bytes (*C function*), 91

function), 93
 wally_scriptpubkey_to_address (C function), 20
 wally_scriptsig_multisig_from_bytes (C function), 94
 wally_scriptsig_p2pkh_from_der (C function), 92
 wally_scriptsig_p2pkh_from_sig (C function), 91
 wally_scrip (C function), 7
 wally_secp_context_free (C function), 1
 wally_secp_randomize (C function), 2
 wally_set_operations (C function), 5
 wally_sha256 (C function), 8
 wally_sha256_midstate (C function), 8
 wally_sha256d (C function), 8
 wally_sha512 (C function), 9
 wally_symmetric_key_from_parent (C function), 107
 wally_symmetric_key_from_seed (C function), 107
 wally_tx_add_elements_raw_input (C function), 123
 wally_tx_add_elements_raw_input_at (C function), 124
 wally_tx_add_elements_raw_output (C function), 125
 wally_tx_add_elements_raw_output_at (C function), 126
 wally_tx_add_input (C function), 112
 wally_tx_add_input_at (C function), 112
 wally_tx_add_output (C function), 114
 wally_tx_add_output_at (C function), 114
 wally_tx_add_raw_input (C function), 113
 wally_tx_add_raw_input_at (C function), 113
 wally_tx_add_raw_output (C function), 114
 wally_tx_add_raw_output_at (C function), 115
 wally_tx_clone_alloc (C function), 112
 wally_tx_confidential_value_from_satoshii (C function), 127
 wally_tx_confidential_value_to_satoshii (C function), 127
 wally_tx_elements_input_init_alloc (C function), 120
 wally_tx_elements_input_is_pegin (C function), 121
 wally_tx_elements_input_issuance_free (C function), 120
 wally_tx_elements_input_issuance_set (C function), 119
 wally_tx_elements_issuance_calculate_asset (C function), 128
 wally_tx_elements_issuance_calculate_reissuance (C function), 128
 wally_tx_elements_issuance_generate_entropy (C function), 128
 wally_tx_elements_output_commitment_free (C function), 122
 wally_tx_elements_output_commitment_set (C function), 121
 wally_tx_elements_output_init (C function), 122
 wally_tx_elements_output_init_alloc (C function), 123
 wally_tx_free (C function), 115
 wally_tx_from_bytes (C function), 116
 wally_tx_from_hex (C function), 116
 wally_tx_get_btc_signature_hash (C function), 117
 wally_tx_get_btc_taproot_signature_hash (C function), 118
 wally_tx_get_elements_signature_hash (C function), 127
 wally_tx_get_length (C function), 116
 wally_tx_get_signature_hash (C function), 118
 wally_tx_get_total_output_satoshii (C function), 117
 wally_tx_get_txid (C function), 115
 wally_tx_get_vsize (C function), 117
 wally_tx_get_weight (C function), 117
 wally_tx_get_witness_count (C function), 115
 wally_tx_init_alloc (C function), 112
 wally_tx_input_free (C function), 111
 wally_tx_input_init_alloc (C function), 110
 wally_tx_is_coinbase (C function), 119
 wally_tx_is_elements (C function), 127
 wally_tx_output_clone (C function), 111
 wally_tx_output_clone_alloc (C function), 111
 wally_tx_output_free (C function), 112
 wally_tx_output_init (C function), 111
 wally_tx_output_init_alloc (C function), 111
 wally_tx_remove_input (C function), 113
 wally_tx_remove_output (C function), 115
 wally_tx_set_input_script (C function), 114
 wally_tx_set_input_witness (C function), 114
 wally_tx_to_bytes (C function), 116
 wally_tx_to_hex (C function), 116
 wally_tx_vsize_from_weight (C function), 117
 wally_tx_witness_stack_add (C function), 109
 wally_tx_witness_stack_add_dummy (C function), 109
 wally_tx_witness_stack_clone_alloc (C function), 109
 wally_tx_witness_stack_free (C function), 110

wally_tx_witness_stack_init_alloc (C
function), 109

wally_tx_witness_stack_set (C function), 110

wally_tx_witness_stack_set_dummy (C func-
tion), 110

wally_varbuff_get_length (C function), 97

wally_varbuff_to_bytes (C function), 97

wally_varint_get_length (C function), 96

wally_varint_to_bytes (C function), 96

wally_wif_from_bytes (C function), 21

wally_wif_is_uncompressed (C function), 21

wally_wif_to_address (C function), 22

wally_wif_to_bytes (C function), 21

wally_wif_to_public_key (C function), 21

wally_witness_multisig_from_bytes (C
function), 95

wally_witness_p2wpkh_from_der (C function),
93

wally_witness_p2wpkh_from_sig (C function),
92

wally_witness_program_from_bytes (C func-
tion), 97

wally_witness_program_from_bytes_and_version
(C function), 98