libwally-core Documentation

Release 0.7.3

Jon Griffiths

Jul 18, 2019

Contents:

CHAPTER 1

Core Functions

int **wally_init**(uint32_t *flags*)
Initialize wally.

As wally is not currently threadsafe, this function should be called once before threads are created by the application.

Parameters

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

Returns WALLY_OK or an error code.

Return type int

int **wally_cleanup**(uint32_t *flags*)
Free any internally allocated memory.

Parameters

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

Returns WALLY_OK or an error code.

Return type int

struct secp256k1_context_struct ***wally_get_secp_context**(void)
Fetch the wally internal secp256k1 context object.

The context is created on demand.

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 WALLY_OK or an error code.

Return type int

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 WALLY_OK or an error code.

Return type int

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. 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).

As wally is not currently threadsafe, 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 WALLY_OK or an error code.

Return type int

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 WALLY_OK or an error code.

Return type int

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 WALLY_OK or an error code.

Return type int

int **wally_base58_from_bytes**(const unsigned char *_bytes_, size_t _bytes_len_, uint32_t _flags_, char **_output_)

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 WALLY_OK or an error code.

Return type int

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 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 WALLY_OK or an error code.

Return type int

int **wally_base58_get_length**(const char *_str_in_, size_t *_written_)

Return the length of a base58 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 WALLY_OK or an error code.

Return type int

int **wally_get_operations**(struct wally_operations *_output_)

Fetch the current overridable operations used by wally.

Parameters

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

Returns WALLY_OK or an error code.

Return type int

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

Set the current overridable operations used by wally.

Parameters

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

Returns WALLY_OK or an error code.

Return type int

int **wally_is_elements_build**(uint64_t *value_out)

Determine if the library was built with elements support.

Parameters

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

Returns WALLY_OK or an error code.

Return type int

CHAPTER 2

Crypto Functions

int **wally_scrypt**(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.

Returns WALLY_OK or an error code.

Return type int

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 WALLY_OK or an error code.

Return type int

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

Returns WALLY_OK or an error code.

Return type int

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** – The length of bytes_out in bytes. Must be SHA256_LEN.

Returns WALLY_OK or an error code.

Return type int

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** – The length of `bytes_out` in bytes. Must be `SHA256_LEN`.

Returns WALLY_OK or an error code.

Return type int

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** – The length of `bytes_out` in bytes. Must be `SHA256_LEN`.

Returns WALLY_OK or an error code.

Return type int

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** – The length of `bytes_out` in bytes. Must be `SHA512_LEN`.

Returns WALLY_OK or an error code.

Return type int

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** – The length of `bytes_out` in bytes. Must be `HASH160_LEN`.

Returns WALLY_OK or an error code.

Return type int

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** – The length of `bytes_out` in bytes. Must be `HMAC_SHA256_LEN`.

Returns WALLY_OK or an error code.

Return type int

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** – The length of `bytes_out` in bytes. Must be `HMAC_SHA512_LEN`.

Returns WALLY_OK or an error code.

Return type int

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** – The length of `bytes_out` in bytes. This must be a multiple of `PBKDF2_HMAC_SHA256_LEN`.

Returns WALLY_OK or an error code.

Return type int

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** – The length of bytes_out in bytes. This must be a multiple of PBKDF2_HMAC_SHA512_LEN.

Returns WALLY_OK or an error code.

Return type int

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 WALLY_OK or an error code.

Return type int

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 WALLY_OK or an error code.

Return type int

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** – The length of bytes_out in bytes. Must be EC_PUBLIC_KEY_LEN.

Returns WALLY_OK or an error code.

Return type int

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** – The length of `bytes_out` in bytes. Must be `EC_PUBLIC_KEY_UNCOMPRESSED_LEN`.

Returns WALLY_OK or an error code.

Return type int

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_LEN`.

Returns WALLY_OK or an error code.

Return type int

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** – The length of `bytes_out` in bytes. Must be `EC_SIGNATURE_LEN`.

Returns WALLY_OK or an error code.

Return type int

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.
- **len** – The length of `bytes_out` in bytes. Must be `EC_SIGNATURE_DER_MAX_LEN`.
- **written** – Destination for the number of bytes written to `bytes_out`.

Returns WALLY_OK or an error code.

Return type int

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** – The length of `bytes_out` in bytes. Must be `EC_SIGNATURE_LEN`.

Returns WALLY_OK or an error code.

Return type int

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 WALLY_OK or an error code.

Return type int

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 WALLY_OK or an error code.

Return type int

int **wally_ecdh**(const unsigned char **pub_key*, size_t *pub_key_len*, const unsigned char **bytes*, size_t *bytes_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 `pubkey` in bytes. Must be `EC_PUBLIC_KEY_LEN`.
- **bytes** – The private key.
- **bytes_len** – The length of `privkey` in bytes. Must be `EC_PRIVATE_KEY_LEN`.
- **output** – Destination for the shared secret.
- **output_len** – The length of `output` in bytes. Must be `SHA256_LEN`.

Returns WALLY_OK or an error code.

Return type int

CHAPTER 3

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 witness program.

Parameters

- **bytes** – Witness program bytes, including the version and data push opcode.
- **bytes_len** – Length of `bytes` in bytes. Must be 20 or 32 if script_version is 0.
- **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 WALLY_OK or an error code.

Return type int

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 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 witness program bytes.
- **len** – Length of `bytes_out` in bytes.
- **written** – Destination for the number of bytes written to `bytes_out`.

Returns WALLY_OK or an error code.

Return type int

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** – Prefix byte to use, e.g. 0x80, 0xef.
- **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 WALLY_OK or an error code.

Return type int

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. 0x80, 0xef.
- **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** – The length of `bytes_out` in bytes. Must be `EC_PRIVATE_KEY_LEN`.

Returns WALLY_OK or an error code.

Return type int

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 WALLY_OK or an error code.

Return type int

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 WALLY_OK or an error code.

Return type int

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. 0x00, 0x6f.
- **output** – Destination for the resulting address string.

Returns WALLY_OK or an error code.

Return type int

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 WALLY_OK or an error code.

Return type int

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** – The length of `bytes_out` in bytes. Must be `EC_PUBLIC_KEY_LEN`.

Returns WALLY_OK or an error code.

Return type int

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 WALLY_OK or an error code.

Return type int

CHAPTER 4

Bip32 Functions

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

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

Parameters

- **hdkey** – Key to free.

Returns WALLY_OK or an error code.

Return type int

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*)

Returns WALLY_OK or an error code.

Return type int

int **bip32_key_from_seed**(const unsigned char **bytes*, size_t *bytes_len*, uint32_t *version*, 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.
- **flags** – Either `BIP32_FLAG_SKIP_HASH` to skip hash160 calculation, or 0.

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

Returns WALLY_OK or an error code.

Return type int

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 WALLY_OK or an error code.

Return type int

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` in bytes. Must be `BIP32_SERIALIZED_LEN`.

Returns WALLY_OK or an error code.

Return type int

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 WALLY_OK or an error code.

Return type int

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 WALLY_OK or an error code.

Return type int

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 WALLY_OK or an error code.

Return type int

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 WALLY_OK or an error code.

Return type int

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_KEY_** Flags indicating the type of derivation wanted.
- **output** – Destination for the resulting child extended key.

Returns WALLY_OK or an error code.

Return type int

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 WALLY_OK or an error code.

Return type int

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.

Returns WALLY_OK or an error code.

Return type int

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 WALLY_OK or an error code.

Return type int

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 WALLY_OK or an error code.

Return type int

CHAPTER 5

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` in bytes. Must be `BIP38_SERIALIZED_LEN`.

Returns WALLY_OK or an error code.

Return type int

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 WALLY_OK or an error code.

Return type int

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` in bytes. Must be `EC_PRIVATE_KEY_LEN`.

Returns WALLY_OK or an error code.

Return type int

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` in bytes. Must be `EC_PRIVATE_KEY_LEN`.

Returns WALLY_OK or an error code.

Return type int

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 WALLY_OK or an error code.

Return type int

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 WALLY_OK or an error code.

Return type int

CHAPTER 6

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*. :return: WALLY_OK or an error code. :rtype: int

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 WALLY_OK or an error code.

Return type int

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*. :return: WALLY_OK or an error code. :rtype: int

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 WALLY_OK or an error code.

Return type int

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.
- **len** – The length of `bytes_out` in bytes.
- **written** – Destination for the number of bytes written to `bytes_out`.

Returns WALLY_OK or an error code.

Return type int

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 WALLY_OK or an error code.

Return type int

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** – The length of `bytes_out` in bytes. Currently This must be `BIP39_SEED_LEN_512`.
- **written** – Destination for the number of bytes written to `bytes_out`.

Returns WALLY_OK or an error code.

Return type int

CHAPTER 7

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 WALLY_OK or an error code.

Return type int

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

Returns WALLY_OK or an error code.

Return type int

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

Returns WALLY_OK or an error code.

Return type int

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` `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.
- **len** – The length of `bytes_out` in bytes.
- **written** – Destination for the number of bytes written to `bytes_out`.

Returns WALLY_OK or an error code.

Return type int

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.
- **len** – The length of `bytes_out` in bytes. Passing `WALLY_SCRIPTPUBKEY_OP_RETURN_MAX_LEN` will ensure there is always enough room for the resulting scriptPubkey.
- **written** – Destination for the number of bytes written to `bytes_out`.

Returns WALLY_OK or an error code.

Return type int

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.
- **bytes_len** – Length of `bytes` in bytes.
- **flags** – `WALLY_SCRIPT_HASH160` or 0.
- **bytes_out** – Destination for the resulting scriptPubkey.
- **len** – The length of `bytes_out` in bytes. If `WALLY_SCRIPT_HASH160` is given, `bytes` is a script to hash160 before creating the P2SH. Otherwise, bytes_len must be `HASH160_LEN` and `bytes` must contain the hash160 to use.
- **written** – Destination for the number of bytes written to `bytes_out`.

Returns WALLY_OK or an error code.

Return type int

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 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 WALLY_OK or an error code.

Return type int

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 WALLY_OK or an error code.

Return type int

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 non-zero and less than 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 WALLY_OK or an error code.

Return type int

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 non-zero and less than 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 WALLY_OK or an error code.

Return type int

int **wally_script_push_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 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 WALLY_OK or an error code.

Return type int

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.
- **len** – The length of bytes_out in bytes.
- **written** – Destination for the number of bytes written to bytes_out.

Returns WALLY_OK or an error code.

Return type int

CHAPTER 8

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 WALLY_OK or an error code.

Return type int

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 WALLY_OK or an error code.

Return type int

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 WALLY_OK or an error code.

Return type int

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 WALLY_OK or an error code.

Return type int

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 WALLY_OK or an error code.

Return type int

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 WALLY_OK or an error code.

Return type int

int **wally_tx_input_init_alloc**(const unsigned char **txhash*, size_t *txhash_len*, uint32_t *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`.
- **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 WALLY_OK or an error code.

Return type int

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 WALLY_OK or an error code.

Return type int

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.

:param satoshi The amount of the output in satoshi. :param script: The scriptPubkey for the output. :param script_len: Size of `script` in bytes. :param output: Destination for the resulting transaction output. :return: WALLY_OK or an error code. :rtype: int

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 WALLY_OK or an error code.

Return type int

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 WALLY_OK or an error code.

Return type int

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 WALLY_OK or an error code.

Return type int

int **wally_tx_add_raw_input**(struct wally_tx **tx*, const unsigned char **txhash*, size_t *txhash_len*, uint32_t *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`.
- **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 script creation. Must be 0.

Returns WALLY_OK or an error code.

Return type int

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 WALLY_OK or an error code.

Return type int

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 WALLY_OK or an error code.

Return type int

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 WALLY_OK or an error code.

Return type int

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 input to.
- **output** – The transaction output to add to `tx`.

Returns WALLY_OK or an error code.

Return type int

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 input 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 script creation. Must be 0.

Returns WALLY_OK or an error code.

Return type int

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 WALLY_OK or an error code.

Return type int

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 WALLY_OK or an error code.

Return type int

int **wally_tx_free**(struct wally_tx **tx*)
Free a transaction allocated by *wally_tx_init_alloc*.

Parameters

- **tx** – The transaction to free.

Returns WALLY_OK or an error code.

Return type int

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 WALLY_OK or an error code.

Return type int

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 WALLY_OK or an error code.

Return type int

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 WALLY_OK or an error code.

Return type int

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 WALLY_OK or an error code.

Return type int

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 WALLY_OK or an error code.

Return type int

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 WALLY_OK or an error code.

Return type int

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 WALLY_OK or an error code.

Return type int

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 WALLY_OK or an error code.

Return type int

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 WALLY_OK or an error code.

Return type int

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 scriptPubkey spent by the input being signed for.
- **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` in bytes. Must be at least `SHA256_LEN`.

Returns WALLY_OK or an error code.

Return type int

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 scriptPubkey spent by the input being signed for.
- **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 endianess; 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` in bytes. Must be at least `SHA256_LEN`.

Returns WALLY_OK or an error code.

Return type int

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 WALLY_OK or an error code.

Return type int

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 WALLY_OK or an error code.

Return type int

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 WALLY_OK or an error code.

Return type int

int **wally_tx_elements_input_init_alloc**(const unsigned char **txhash*, size_t *txhash_len*, uint32_t *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 **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*, 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`.
- **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 WALLY_OK or an error code.

Return type int

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 WALLY_OK or an error code.

Return type int

int **wally_tx_elements_output_commitment_set**(struct wally_tx_output **input*, 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`.
- **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 WALLY_OK or an error code.

Return type int

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 WALLY_OK or an error code.

Return type int

int **wally_tx_elements_output_init_alloc**(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)

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`.
- **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 WALLY_OK or an error code.

Return type int

int **wally_tx_add_elements_raw_input**(struct wally_tx **tx*, const unsigned char **txhash*, size_t *txhash_len*, uint32_t *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 **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*, 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`.
- **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 WALLY_OK or an error code.

Return type int

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

- **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`.
- **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 WALLY_OK or an error code.

Return type int

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 WALLY_OK or an error code.

Return type int

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` in bytes. Must be `WALLY_TX_ASSET_CT_VALUE_UNBLIND_LEN`.

Returns WALLY_OK or an error code.

Return type int

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 WALLY_OK or an error code.

Return type int

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 a 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 scriptPubkey spent by the input being signed for.
- **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` in bytes. Must be `SHA256_LEN`.

Returns WALLY_OK or an error code.

Return type int

int **wally_tx_elements_issuance_generate_entropy**(const unsigned char **txhash*, size_t *txhash_len*, uint32_t *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 SHA256_LEN.
- **index** – The prevout index.
- **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 in bytes. Must be SHA256_LEN.

Returns WALLY_OK or an error code.

Return type int

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 in bytes. Must be SHA256_LEN.

Returns WALLY_OK or an error code.

Return type int

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.

Parameters

- **bytes_out** – Destination for the re-issuance token.
- **len** – Size of bytes_out in bytes. Must be SHA256_LEN.

Returns WALLY_OK or an error code.

Return type int

CHAPTER 9

Indices and tables

- genindex
- search

Index

B

W