

Nano cryptocurrency C library with P2PoW/DPoW support for Embedded  
1.0.0

Generated by Doxygen 1.8.13



# Contents

<b>1</b>	<b>Overview</b>	<b>1</b>
<b>2</b>	<b>Data Structure Index</b>	<b>3</b>
2.1	Data Structures . . . . .	3
<b>3</b>	<b>File Index</b>	<b>5</b>
3.1	Files . . . . .	5
<b>4</b>	<b>Data Structure Documentation</b>	<b>7</b>
4.1	f_block_transfer_t Struct Reference . . . . .	7
4.1.1	Detailed Description . . . . .	7
4.1.2	Field Documentation . . . . .	7
4.1.2.1	account . . . . .	7
4.1.2.2	balance . . . . .	8
4.1.2.3	link . . . . .	8
4.1.2.4	preamble . . . . .	8
4.1.2.5	prefixes . . . . .	8
4.1.2.6	previous . . . . .	8
4.1.2.7	representative . . . . .	9
4.1.2.8	signature . . . . .	9
4.1.2.9	work . . . . .	9
4.2	f_file_info_err_t Struct Reference . . . . .	9
4.2.1	Detailed Description . . . . .	9
4.3	f_nano_crypto_wallet_t Struct Reference . . . . .	9
4.3.1	Detailed Description . . . . .	10

4.3.2	Field Documentation . . . . .	10
4.3.2.1	description . . . . .	10
4.3.2.2	iv . . . . .	10
4.3.2.3	nano_hdr . . . . .	10
4.3.2.4	salt . . . . .	11
4.3.2.5	seed_block . . . . .	11
4.3.2.6	ver . . . . .	11
4.4	f_nano_encrypted_wallet_t Struct Reference . . . . .	11
4.4.1	Detailed Description . . . . .	11
4.4.2	Field Documentation . . . . .	12
4.4.2.1	hash_sk_unencrypted . . . . .	12
4.4.2.2	iv . . . . .	12
4.4.2.3	reserved . . . . .	12
4.4.2.4	sk_encrypted . . . . .	12
4.4.2.5	sub_salt . . . . .	13
4.5	f_nano_wallet_info_bdy_t Struct Reference . . . . .	13
4.5.1	Detailed Description . . . . .	13
4.5.2	Field Documentation . . . . .	13
4.5.2.1	last_used_wallet_number . . . . .	13
4.5.2.2	max_fee . . . . .	14
4.5.2.3	reserved . . . . .	14
4.5.2.4	wallet_prefix . . . . .	14
4.5.2.5	wallet_representative . . . . .	14
4.6	f_nano_wallet_info_t Struct Reference . . . . .	14
4.6.1	Detailed Description . . . . .	15
4.6.2	Field Documentation . . . . .	15
4.6.2.1	body . . . . .	15
4.6.2.2	desc . . . . .	15
4.6.2.3	file_info_integrity . . . . .	15
4.6.2.4	header . . . . .	16
4.6.2.5	nanoseed_hash . . . . .	16
4.6.2.6	version . . . . .	16

<b>5 File Documentation</b>	<b>17</b>
5.1 f_add_bn_288_le.h File Reference . . . . .	17
5.1.1 Detailed Description . . . . .	17
5.1.2 Typedef Documentation . . . . .	17
5.1.2.1 F_ADD_288 . . . . .	17
5.2 f_add_bn_288_le.h . . . . .	18
5.3 f_nano_crypto_util.h File Reference . . . . .	18
5.3.1 Detailed Description . . . . .	20
5.3.2 Macro Definition Documentation . . . . .	20
5.3.2.1 DEST_XRB . . . . .	21
5.3.2.2 MAX_STR_NANO_CHAR . . . . .	21
5.3.2.3 NANO_ENCRYPTED_SEED_FILE . . . . .	21
5.3.2.4 NANO_FILE_WALLETS_INFO . . . . .	21
5.3.2.5 NANO_PASSWD_MAX_LEN . . . . .	21
5.3.2.6 NANO_PREFIX . . . . .	22
5.3.2.7 PUB_KEY_EXTENDED_MAX_LEN . . . . .	22
5.3.2.8 REP_XRB . . . . .	22
5.3.2.9 SENDER_XRB . . . . .	22
5.3.2.10 STR_NANO_SZ . . . . .	22
5.3.2.11 XRB_PREFIX . . . . .	23
5.3.3 Typedef Documentation . . . . .	23
5.3.3.1 F_FILE_INFO_ERR . . . . .	23
5.3.3.2 f_nano_err . . . . .	23
5.3.3.3 f_uint128_t . . . . .	23
5.3.3.4 f_write_seed_err . . . . .	23
5.3.3.5 NANO_PRIVATE_KEY . . . . .	24
5.3.3.6 NANO_PRIVATE_KEY_EXTENDED . . . . .	24
5.3.3.7 NANO_PUBLIC_KEY . . . . .	24
5.3.3.8 NANO_PUBLIC_KEY_EXTENDED . . . . .	24
5.3.3.9 NANO_SEED . . . . .	24

5.3.4	Enumeration Type Documentation . . . . .	24
5.3.4.1	f_file_info_err_t . . . . .	24
5.3.4.2	f_nano_err_t . . . . .	25
5.3.4.3	f_write_seed_err_t . . . . .	26
5.3.5	Function Documentation . . . . .	26
5.3.5.1	__attribute__() . . . . .	26
5.3.5.2	f_bip39_to_nano_seed() . . . . .	27
5.3.5.3	f_cloud_crypto_wallet_nano_create_seed() . . . . .	28
5.3.5.4	f_generate_nano_seed() . . . . .	29
5.3.5.5	f_get_nano_file_info() . . . . .	29
5.3.5.6	f_nano_add_sub() . . . . .	30
5.3.5.7	f_nano_key_to_str() . . . . .	30
5.3.5.8	f_nano_parse_raw_str_to_raw128_t() . . . . .	32
5.3.5.9	f_nano_parse_real_str_to_raw128_t() . . . . .	32
5.3.5.10	f_nano_raw_to_string() . . . . .	33
5.3.5.11	f_nano_seed_to_bip39() . . . . .	33
5.3.5.12	f_nano_sign_block() . . . . .	34
5.3.5.13	f_nano_transaction_to_JSON() . . . . .	35
5.3.5.14	f_nano_valid_nano_str_value() . . . . .	35
5.3.5.15	f_nano_value_compare_value() . . . . .	36
5.3.5.16	f_nano_verify_nano_funds() . . . . .	37
5.3.5.17	f_parse_nano_seed_and_bip39_to_JSON() . . . . .	37
5.3.5.18	f_read_seed() . . . . .	38
5.3.5.19	f_seed_to_nano_wallet() . . . . .	39
5.3.5.20	f_set_nano_file_info() . . . . .	40
5.3.5.21	is_nano_prefix() . . . . .	40
5.3.5.22	is_null_hash() . . . . .	41
5.3.5.23	nano_base_32_2_hex() . . . . .	41
5.3.5.24	pk_to_wallet() . . . . .	42
5.3.5.25	valid_nano_wallet() . . . . .	42

5.3.5.26	valid_raw_balance()	43
5.3.6	Variable Documentation	43
5.3.6.1	account	43
5.3.6.2	balance	43
5.3.6.3	body	43
5.3.6.4	desc	44
5.3.6.5	description	44
5.3.6.6	file_info_integrity	44
5.3.6.7	hash_sk_unencrypted	44
5.3.6.8	header	44
5.3.6.9	iv	45
5.3.6.10	last_used_wallet_number	45
5.3.6.11	link	45
5.3.6.12	max_fee	45
5.3.6.13	nano_hdr	45
5.3.6.14	nanoseed_hash	46
5.3.6.15	preamble	46
5.3.6.16	prefixes	46
5.3.6.17	previous	46
5.3.6.18	representative	46
5.3.6.19	reserved	47
5.3.6.20	salt	47
5.3.6.21	seed_block	47
5.3.6.22	signature	47
5.3.6.23	sk_encrypted	47
5.3.6.24	sub_salt	48
5.3.6.25	ver	48
5.3.6.26	version	48
5.3.6.27	wallet_prefix	48
5.3.6.28	wallet_representative	48

5.3.6.29	work	49
5.4	f_nano_crypto_util.h	49
5.5	f_util.h File Reference	53
5.5.1	Detailed Description	54
5.5.2	Macro Definition Documentation	54
5.5.2.1	ENTROPY_BEGIN	54
5.5.2.2	ENTROPY_END	54
5.5.2.3	F_ENTROPY_TYPE_EXCELENT	54
5.5.2.4	F_ENTROPY_TYPE_GOOD	55
5.5.2.5	F_ENTROPY_TYPE_NOT_ENOUGH	55
5.5.2.6	F_ENTROPY_TYPE_NOT_RECOMENDED	55
5.5.2.7	F_ENTROPY_TYPE_PARANOIC	55
5.5.2.8	F_PASS_IS_OUT_OVF	56
5.5.2.9	F_PASS_IS_TOO_LONG	56
5.5.2.10	F_PASS_IS_TOO_SHORT	56
5.5.2.11	F_PASS_MUST_HAVE_AT_LEAST_NONE	56
5.5.2.12	F_PASS_MUST_HAVE_AT_LEAST_ONE_NUMBER	56
5.5.2.13	F_PASS_MUST_HAVE_AT_LEAST_ONE_SYMBOL	57
5.5.2.14	F_PASS_MUST_HAVE_AT_LEAST_ONE_UPPER_CASE	57
5.5.3	Typedef Documentation	57
5.5.3.1	rnd_fn	57
5.5.4	Function Documentation	57
5.5.4.1	f_pass_must_have_at_least()	58
5.5.4.2	f_random()	59
5.5.4.3	f_random_attach()	59
5.5.4.4	f_verify_system_entropy()	59
5.5.4.5	get_console_passwd()	60
5.6	f_util.h	60
5.7	sodium.h File Reference	62
5.7.1	Detailed Description	63
5.8	sodium.h	63



# Chapter 1

## Overview

*myNanoEmbedded* is a lightweight C library of source files that integrates Nano Cryptocurrency to low complexity computational devices to send/receive digital money to anywhere in the world with fast transaction and with a small fee by delegating a Proof of Work with your choice:

- DPoW (Distributed Proof of Work)
- P2PoW (a Decentralized P2P Proof of Work)

### API features

- Attaches a random function to TRNG hardware (if available)
- Self entropy verifier to ensure excellent TRNG or PRNG entropy
- Creates an encrypted by password your stream or file to store your Nano SEED
- Bip39 and Brainwallet support
- Convert raw data to Base32
- Parse SEED and Bip39 to JSON
- Sign a block using Blake2b hash with Ed25519 algorithm
- ARM-A, ARM-M, Thumb, Xtensa-LX6 and IA64 compatible
- Linux desktop, Raspberry PI, ESP32 and Olimex A20 tested platforms
- Communication over Fenix protocol bridge over TLS
- Libsodium and mbedTLS libraries with smaller resources and best performance
- Optimized for size and speed
- Non static functions (all data is cleared before processed for security)

### To add this API in your project you must first:

1. Download the latest version.

```
git clone https://github.com/devfabiosilva/myNanoEmbedded.git --recurse-submodules
```

2. Include the main library files in the client application.

```
#include "f_nano_crypto_util.h"
```

### Initialize API

Function	Description
<code>f_random_attach()</code> (p. ??)	Initializes the PRNG or TRNG to be used in this API

## Transmit/Receive transactions

To transmit/receive your transaction you must use `Fenix` protocol to stabilish a DPoW/P2PoW support

## Examples using platforms

The repository has some examples with most common embedded and Linux systems

- Native Linux
- Raspberry Pi
- ESP32
- Olimex A20
- STM

## Credits

### Author

Fábio Pereira da Silva

### Date

Feb 2020

### Version

1.0

### Copyright

License MIT [see here](#)

## References:

1- Editing

## Chapter 2

# Data Structure Index

### 2.1 Data Structures

Here are the data structures with brief descriptions:

<b>f_block_transfer_t</b>	
Nano signed block raw data defined in this <a href="#">reference</a> . . . . .	7
<b>f_file_info_err_t</b>	
Error enumerator for info file functions . . . . .	9
<b>f_nano_crypto_wallet_t</b>	
<b>struct</b> of the block of encrypted file to store Nano SEED . . . . .	9
<b>f_nano_encrypted_wallet_t</b>	
<b>struct</b> of the block of encrypted file to store Nano SEED . . . . .	11
<b>f_nano_wallet_info_bdy_t</b>	
<b>struct</b> of the body block of the info file . . . . .	13
<b>f_nano_wallet_info_t</b>	
<b>struct</b> of the body block of the info file . . . . .	14



## Chapter 3

# File Index

### 3.1 Files

Here is a list of all files with brief descriptions:

<b>f_add_bn_288_le.h</b>	
Low level implementation of Nano Cryptocurrency C library . . . . .	17
<b>f_nano_crypto_util.h</b>	
This API Integrates Nano Cryptocurrency to low computational devices . . . . .	18
<b>f_util.h</b>	
This ABI is a utility for myNanoEmbedded library and sub routines are implemented here . . .	53
<b>sodium.h</b>	
This header file is an implementation of Libsodium library . . . . .	62



## Chapter 4

# Data Structure Documentation

### 4.1 `f_block_transfer_t` Struct Reference

```
#include <f_nano_crypto_util.h>
```

#### Data Fields

- `uint8_t` **preamble** [32]
- `uint8_t` **account** [32]
- `uint8_t` **previous** [32]
- `uint8_t` **representative** [32]
- `f_uint128_t` **balance**
- `uint8_t` **link** [32]
- `uint8_t` **signature** [64]
- `uint8_t` **prefixes**
- `uint64_t` **work**

#### 4.1.1 Detailed Description

Nano signed block raw data defined in this [reference](#)

Definition at line **239** of file **`f_nano_crypto_util.h`**.

#### 4.1.2 Field Documentation

##### 4.1.2.1 `account`

```
uint8_t account[32]
```

Account in raw binary data.

Definition at line **243** of file **`f_nano_crypto_util.h`**.

#### 4.1.2.2 balance

`f_uint128_t balance`

Big number 128 bit raw balance.

See also

`f_uint128_t` (p. ??)

Definition at line 251 of file `f_nano_crypto_util.h`.

#### 4.1.2.3 link

`uint8_t link[32]`

link or destination account

Definition at line 253 of file `f_nano_crypto_util.h`.

#### 4.1.2.4 preamble

`uint8_t preamble[32]`

Block preamble.

Definition at line 241 of file `f_nano_crypto_util.h`.

#### 4.1.2.5 prefixes

`uint8_t prefixes`

Internal use for this API.

Definition at line 257 of file `f_nano_crypto_util.h`.

#### 4.1.2.6 previous

`uint8_t previous[32]`

Previous block.

Definition at line 245 of file `f_nano_crypto_util.h`.



#### 4.1.2.7 `representative`

```
uint8_t representative[32]
```

Representative for current account.

Definition at line **247** of file `f_nano_crypto_util.h`.

#### 4.1.2.8 `signature`

```
uint8_t signature[64]
```

Signature of the block.

Definition at line **255** of file `f_nano_crypto_util.h`.

#### 4.1.2.9 `work`

```
uint64_t work
```

Internal use for this API.

Definition at line **259** of file `f_nano_crypto_util.h`.

The documentation for this struct was generated from the following file:

- `f_nano_crypto_util.h`

## 4.2 `f_file_info_err_t` Struct Reference

```
#include <f_nano_crypto_util.h>
```

### 4.2.1 Detailed Description

Error enumerator for info file functions.

The documentation for this struct was generated from the following file:

- `f_nano_crypto_util.h`

## 4.3 `f_nano_crypto_wallet_t` Struct Reference

```
#include <f_nano_crypto_util.h>
```

## Data Fields

- `uint8_t nano_hdr` [`sizeof(NANO_WALLET_MAGIC)`]
- `uint32_t ver`
- `uint8_t description` [`F_DESC_SZ`]
- `uint8_t salt` [32]
- `uint8_t iv` [16]
- `F_ENCRYPTED_BLOCK seed_block`

### 4.3.1 Detailed Description

**struct** of the block of encrypted file to store Nano SEED

Definition at line **370** of file **f\_nano\_crypto\_util.h**.

### 4.3.2 Field Documentation

#### 4.3.2.1 description

```
uint8_t description[F_DESC_SZ]
```

File description.

Definition at line **376** of file **f\_nano\_crypto\_util.h**.

#### 4.3.2.2 iv

```
uint8_t iv[16]
```

Initial vector of first encryption layer.

Definition at line **380** of file **f\_nano\_crypto\_util.h**.

#### 4.3.2.3 nano\_hdr

```
uint8_t nano_hdr[sizeof(NANO_WALLET_MAGIC)]
```

Header of the file.

Definition at line **372** of file **f\_nano\_crypto\_util.h**.

#### 4.3.2.4 salt

```
uint8_t salt[32]
```

Salt of the first encryption layer.

Definition at line 378 of file `f_nano_crypto_util.h`.

#### 4.3.2.5 seed\_block

```
F_ENCRYPTED_BLOCK seed_block
```

Second encrypted block for Nano SEED.

Definition at line 382 of file `f_nano_crypto_util.h`.

#### 4.3.2.6 ver

```
uint32_t ver
```

Version of the file.

Definition at line 374 of file `f_nano_crypto_util.h`.

The documentation for this struct was generated from the following file:

- `f_nano_crypto_util.h`

## 4.4 f\_nano\_encrypted\_wallet\_t Struct Reference

```
#include <f_nano_crypto_util.h>
```

### Data Fields

- `uint8_t sub_salt` [32]
- `uint8_t iv` [16]
- `uint8_t reserved` [16]
- `uint8_t hash_sk_unencrypted` [32]
- `uint8_t sk_encrypted` [32]

#### 4.4.1 Detailed Description

**struct** of the block of encrypted file to store Nano SEED

Definition at line 342 of file `f_nano_crypto_util.h`.

## 4.4.2 Field Documentation

### 4.4.2.1 hash\_sk\_unencrypted

```
uint8_t hash_sk_unencrypted[32]
```

hash of Nano SEED when unencrypted

Definition at line **350** of file **f\_nano\_crypto\_util.h**.

### 4.4.2.2 iv

```
uint8_t iv[16]
```

Initial sub vector.

Definition at line **346** of file **f\_nano\_crypto\_util.h**.

### 4.4.2.3 reserved

```
uint8_t reserved[16]
```

Reserved (not used)

Definition at line **348** of file **f\_nano\_crypto\_util.h**.

### 4.4.2.4 sk\_encrypted

```
uint8_t sk_encrypted[32]
```

Secret.

SEED encrypted (second layer)

Definition at line **352** of file **f\_nano\_crypto\_util.h**.

#### 4.4.2.5 sub\_salt

```
uint8_t sub_salt[32]
```

Salt of the sub block to be stored.

Definition at line **344** of file **f\_nano\_crypto\_util.h**.

The documentation for this struct was generated from the following file:

- **f\_nano\_crypto\_util.h**

## 4.5 f\_nano\_wallet\_info\_bdy\_t Struct Reference

```
#include <f_nano_crypto_util.h>
```

### Data Fields

- uint8\_t **wallet\_prefix**
- uint32\_t **last\_used\_wallet\_number**
- char **wallet\_representative** [ MAX\_STR\_NANO\_CHAR]
- char **max\_fee** [F\_RAW\_STR\_MAX\_SZ]
- uint8\_t **reserved** [44]

### 4.5.1 Detailed Description

**struct** of the body block of the info file

Definition at line **454** of file **f\_nano\_crypto\_util.h**.

### 4.5.2 Field Documentation

#### 4.5.2.1 last\_used\_wallet\_number

```
uint32_t last_used_wallet_number
```

Last used wallet number.

Definition at line **458** of file **f\_nano\_crypto\_util.h**.

#### 4.5.2.2 max\_fee

```
char max_fee[F_RAW_STR_MAX_SZ]
```

Custom preferred max fee of Proof of Work.

Definition at line **462** of file **f\_nano\_crypto\_util.h**.

#### 4.5.2.3 reserved

```
uint8_t reserved[44]
```

Reserved.

Definition at line **464** of file **f\_nano\_crypto\_util.h**.

#### 4.5.2.4 wallet\_prefix

```
uint8_t wallet_prefix
```

Wallet prefix: 0 for NANO; 1 for XRB.

Definition at line **456** of file **f\_nano\_crypto\_util.h**.

#### 4.5.2.5 wallet\_representative

```
char wallet_representative[ MAX_STR_NANO_CHAR]
```

Wallet representative.

Definition at line **460** of file **f\_nano\_crypto\_util.h**.

The documentation for this struct was generated from the following file:

- **f\_nano\_crypto\_util.h**

## 4.6 f\_nano\_wallet\_info\_t Struct Reference

```
#include <f_nano_crypto_util.h>
```

## Data Fields

- `uint8_t header` [`sizeof(F_NANO_WALLET_INFO_MAGIC)`]
- `uint16_t version`
- `char desc` [`F_NANO_DESC_SZ`]
- `uint8_t nanoseed_hash` [`32`]
- `uint8_t file_info_integrity` [`32`]
- `F_NANO_WALLET_INFO_BODY body`

### 4.6.1 Detailed Description

**struct** of the body block of the info file

Definition at line **486** of file `f_nano_crypto_util.h`.

### 4.6.2 Field Documentation

#### 4.6.2.1 `body`

```
F_NANO_WALLET_INFO_BODY body
```

Body of the file info.

Definition at line **498** of file `f_nano_crypto_util.h`.

#### 4.6.2.2 `desc`

```
char desc[F_NANO_DESC_SZ]
```

Description.

Definition at line **492** of file `f_nano_crypto_util.h`.

#### 4.6.2.3 `file_info_integrity`

```
uint8_t file_info_integrity[32]
```

File info integrity of the body block.

Definition at line **496** of file `f_nano_crypto_util.h`.

#### 4.6.2.4 header

```
uint8_t header[sizeof(F_NANO_WALLET_INFO_MAGIC)]
```

Header magic.

Definition at line **488** of file **f\_nano\_crypto\_util.h**.

#### 4.6.2.5 nanoseed\_hash

```
uint8_t nanoseed_hash[32]
```

Nano SEED hash file.

Definition at line **494** of file **f\_nano\_crypto\_util.h**.

#### 4.6.2.6 version

```
uint16_t version
```

Version.

Definition at line **490** of file **f\_nano\_crypto\_util.h**.

The documentation for this struct was generated from the following file:

- **f\_nano\_crypto\_util.h**



## Chapter 5

# File Documentation

### 5.1 `f_add_bn_288_le.h` File Reference

```
#include <stdint.h>
```

#### Typedefs

- typedef uint8\_t **F\_ADD\_288**[36]

#### 5.1.1 Detailed Description

Low level implementation of Nano Cryptocurrency C library.

Definition in file `f_add_bn_288_le.h`.

#### 5.1.2 Typedef Documentation

##### 5.1.2.1 `F_ADD_288`

`F_ADD_288`

288 bit big number

Definition at line **19** of file `f_add_bn_288_le.h`.

## 5.2 f\_add\_bn\_288\_le.h

```

00001  /*
00002      AUTHOR: Fábio Pereira da Silva
00003      YEAR: 2019-20
00004      LICENSE: MIT
00005      EMAIL: fabioegel@gmail.com or fabioegel@protonmail.com
00006  */
00007
00008  #include <stdint.h>
00009
00019  typedef uint8_t F_ADD_288[36];
00020
00021
00022  #ifndef F_DOC_SKIP
00023
00033  void f_add_bn_288_le(F_ADD_288, F_ADD_288, F_ADD_288, int *, int);
00034  void f_sl_elv_add_le(F_ADD_288, int);
00035
00036  #endif
00037

```

## 5.3 f\_nano\_crypto\_util.h File Reference

```

#include <stdint.h>
#include "f_util.h"

```

### Data Structures

- struct **f\_block\_transfer\_t**
- struct **f\_nano\_encrypted\_wallet\_t**
- struct **f\_nano\_crypto\_wallet\_t**
- struct **f\_nano\_wallet\_info\_bdy\_t**
- struct **f\_nano\_wallet\_info\_t**

### Macros

- #define **MAX\_STR\_NANO\_CHAR** (size\_t)70
- #define **PUB\_KEY\_EXTENDED\_MAX\_LEN** (size\_t)40
- #define **NANO\_PREFIX** "nano\_"
- #define **XRB\_PREFIX** "xrb\_"
- #define **NANO\_ENCRYPTED\_SEED\_FILE** "/spiffs/secure/nano.nse"
- #define **NANO\_PASSWD\_MAX\_LEN** (size\_t)80
- #define **STR\_NANO\_SZ** (size\_t)66
- #define **NANO\_FILE\_WALLETS\_INFO** "/spiffs/secure/walletsinfo.i"
- #define **REP\_XRB** (uint8\_t)0x4
- #define **SENDER\_XRB** (uint8\_t)0x02
- #define **DEST\_XRB** (uint8\_t)0x01

### Typedefs

- typedef uint8\_t **NANO\_SEED**[crypto\_sign\_SEEDBYTES]
- typedef uint8\_t **f\_uint128\_t**[16]
- typedef uint8\_t **NANO\_PRIVATE\_KEY**[sizeof( **NANO\_SEED**)]
- typedef uint8\_t **NANO\_PRIVATE\_KEY\_EXTENDED**[crypto\_sign\_ed25519\_SECRETKEYBYTES]
- typedef uint8\_t **NANO\_PUBLIC\_KEY**[crypto\_sign\_ed25519\_PUBLICKEYBYTES]
- typedef uint8\_t **NANO\_PUBLIC\_KEY\_EXTENDED**[**PUB\_KEY\_EXTENDED\_MAX\_LEN**]
- typedef enum **f\_nano\_err\_t** **f\_nano\_err**
- typedef enum **f\_write\_seed\_err\_t** **f\_write\_seed\_err**
- typedef enum **f\_file\_info\_err\_t** **F\_FILE\_INFO\_ERR**

## Enumerations

- enum **f\_nano\_err\_t** {  
**NANO\_ERR\_OK** =0, **NANO\_ERR\_CANT\_PARSE\_BN\_STR** =5151, **NANO\_ERR\_MALLOC**, **NANO\_ERR\_CANT\_PARSE\_FACTOR**,  
**NANO\_ERR\_MPI\_MULT**, **NANO\_ERR\_CANT\_PARSE\_TO\_BLK\_TRANSFER**, **NANO\_ERR\_EMPTY\_STR**, **NANO\_ERR\_CANT\_PARSE\_VALUE**,  
**NANO\_ERR\_PARSE\_MPI\_TO\_STR**, **NANO\_ERR\_CANT\_COMPLETE\_NULL\_CHAR**, **NANO\_ERR\_CANT\_PARSE\_TO\_MPI**, **NANO\_ERR\_INSUFFICIENT\_FUNDS**,  
**NANO\_ERR\_SUB\_MPI**, **NANO\_ERR\_ADD\_MPI**, **NANO\_ERR\_NO\_SENSE\_VALUE\_TO\_SEND\_NEGATIVE**, **NANO\_ERR\_NO\_SENSE\_VALUE\_TO\_SEND\_ZERO**,  
**NANO\_ERR\_NO\_SENSE\_BALANCE\_NEGATIVE**, **NANO\_ERR\_VAL\_A\_INVALID\_MODE**, **NANO\_ERR\_CANT\_PARSE\_TO\_TEMP\_UINT128\_T**, **NANO\_ERR\_VAL\_B\_INVALID\_MODE**,  
**NANO\_ERR\_CANT\_PARSE\_RAW\_A\_TO\_MPI**, **NANO\_ERR\_CANT\_PARSE\_RAW\_B\_TO\_MPI**, **NANO\_ERR\_UNKNOWN\_ADD\_SUB\_MODE**, **NANO\_ERR\_INVALID\_RES\_OUTPUT** }
- enum **f\_write\_seed\_err\_t** {  
**WRITE\_ERR\_OK** =0, **WRITE\_ERR\_NULL\_PASSWORD** =7180, **WRITE\_ERR\_EMPTY\_STRING**, **WRITE\_ERR\_MALLOC**,  
**WRITE\_ERR\_ENCRYPT\_PRIV\_KEY**, **WRITE\_ERR\_GEN\_SUB\_PRIV\_KEY**, **WRITE\_ERR\_GEN\_MAIN\_PRIV\_KEY**, **WRITE\_ERR\_ENCRYPT\_SUB\_BLOCK**,  
**WRITE\_ERR\_UNKNOWN\_OPTION**, **WRITE\_ERR\_FILE\_ALREADY\_EXISTS**, **WRITE\_ERR\_CREATING\_FILE**, **WRITE\_ERR\_WRITING\_FILE** }
- enum **f\_file\_info\_err\_t** {  
**F\_FILE\_INFO\_ERR\_OK** =0, **F\_FILE\_INFO\_ERR\_CANT\_OPEN\_INFO\_FILE** =7001, **F\_FILE\_INFO\_ERR\_NANO\_SEED\_ENCRYPTED\_FILE\_NOT\_FOUND**, **F\_FILE\_INFO\_ERR\_CANT\_DELETE\_NANO\_INFO\_FILE**,  
**F\_FILE\_INFO\_ERR\_MALLOC**, **F\_FILE\_INFO\_ERR\_CANT\_READ\_NANO\_SEED\_ENCRYPTED\_FILE**, **F\_FILE\_INFO\_ERR\_CANT\_READ\_INFO\_FILE**, **F\_FILE\_INFO\_INVALID\_HEADER\_FILE**,  
**F\_FILE\_INFO\_ERR\_INVALID\_SHA256\_INFO\_FILE**, **F\_FILE\_INFO\_ERR\_NANO\_SEED\_HASH\_FAIL**, **F\_FILE\_INFO\_ERR\_NANO\_INVALID\_REPRESENTATIVE**, **F\_FILE\_INFO\_ERR\_NANO\_INVALID\_MAX\_FEE\_VALUE**,  
**F\_FILE\_INFO\_ERR\_OPEN\_FOR\_WRITE\_INFO**, **F\_FILE\_INFO\_ERR\_EXISTING\_FILE**, **F\_FILE\_INFO\_ERR\_CANT\_WRITE\_FILE\_INFO** }

## Functions

- struct **f\_block\_transfer\_t** **\_\_attribute\_\_((packed))** **F\_BLOCK\_TRANSFER**
- int **f\_cloud\_crypto\_wallet\_nano\_create\_seed** (size\_t, char \*, char \*)
- int **f\_generate\_nano\_seed** ( **NANO\_SEED**, uint32\_t)
- int **pk\_to\_wallet** (char \*, char \*, **NANO\_PUBLIC\_KEY\_EXTENDED**)
- int **f\_seed\_to\_nano\_wallet** ( **NANO\_PRIVATE\_KEY**, **NANO\_PUBLIC\_KEY**, **NANO\_SEED**, uint32\_t)
- char \* **f\_nano\_key\_to\_str** (char \*, unsigned char \*)
- int **f\_nano\_seed\_to\_bip39** (char \*, size\_t, size\_t \*, **NANO\_SEED**, char \*)
- int **f\_bip39\_to\_nano\_seed** (uint8\_t \*, char \*, char \*)
- int **f\_parse\_nano\_seed\_and\_bip39\_to\_JSON** (char \*, size\_t, size\_t \*, void \*, int, const char \*)
- int **f\_read\_seed** (uint8\_t \*, const char \*, void \*, int, int)
- int **f\_nano\_raw\_to\_string** (char \*, size\_t \*, size\_t, void \*, int)
- int **f\_nano\_valid\_nano\_str\_value** (const char \*)
- int **valid\_nano\_wallet** (const char \*)
- int **nano\_base\_32\_2\_hex** (uint8\_t \*, char \*)
- int **f\_nano\_transaction\_to\_JSON** (char \*, size\_t, size\_t \*, **NANO\_PRIVATE\_KEY\_EXTENDED**, **F\_BLOCK\_TRANSFER** \*)
- int **valid\_raw\_balance** (const char \*)
- int **is\_null\_hash** (uint8\_t \*)
- int **is\_nano\_prefix** (const char \*, const char \*)
- **F\_FILE\_INFO\_ERR** **f\_get\_nano\_file\_info** (**F\_NANO\_WALLET\_INFO** \*)

- **F\_FILE\_INFO\_ERR f\_set\_nano\_file\_info** (F\_NANO\_WALLET\_INFO \*, int)
- **f\_nano\_err f\_nano\_value\_compare\_value** (void \*, void \*, uint32\_t \*)
- **f\_nano\_err f\_nano\_verify\_nano\_funds** (void \*, void \*, void \*, uint32\_t)
- **f\_nano\_err f\_nano\_parse\_raw\_str\_to\_raw128\_t** (uint8\_t \*, const char \*)
- **f\_nano\_err f\_nano\_parse\_real\_str\_to\_raw128\_t** (uint8\_t \*, const char \*)
- **f\_nano\_err f\_nano\_add\_sub** (void \*, void \*, void \*, uint32\_t)
- **int f\_nano\_sign\_block** (F\_BLOCK\_TRANSFER \*, F\_BLOCK\_TRANSFER \*, NANO\_PRIVATE\_KEY\_E←XTENDED)

## Variables

- uint8\_t **preamble** [32]
- uint8\_t **account** [32]
- uint8\_t **previous** [32]
- uint8\_t **representative** [32]
- f\_uint128\_t **balance**
- uint8\_t **link** [32]
- uint8\_t **signature** [64]
- uint8\_t **prefixes**
- uint64\_t **work**
- uint8\_t **sub\_salt** [32]
- uint8\_t **iv** [16]
- uint8\_t **reserved** [16]
- uint8\_t **hash\_sk\_unencrypted** [32]
- uint8\_t **sk\_encrypted** [32]
- uint8\_t **nano\_hdr** [sizeof(NANO\_WALLET\_MAGIC)]
- uint32\_t **ver**
- uint8\_t **description** [F\_DESC\_SZ]
- uint8\_t **salt** [32]
- F\_ENCRYPTED\_BLOCK **seed\_block**
- uint8\_t **wallet\_prefix**
- uint32\_t **last\_used\_wallet\_number**
- char **wallet\_representative** [ MAX\_STR\_NANO\_CHAR]
- char **max\_fee** [F\_RAW\_STR\_MAX\_SZ]
- uint8\_t **header** [sizeof(F\_NANO\_WALLET\_INFO\_MAGIC)]
- uint16\_t **version**
- char **desc** [F\_NANO\_DESC\_SZ]
- uint8\_t **nanoseed\_hash** [32]
- uint8\_t **file\_info\_integrity** [32]
- F\_NANO\_WALLET\_INFO\_BODY **body**

### 5.3.1 Detailed Description

This API Integrates Nano Cryptocurrency to low computational devices.

Definition in file **f\_nano\_crypto\_util.h**.

### 5.3.2 Macro Definition Documentation

#### 5.3.2.1 DEST\_XRB

```
#define DEST_XRB (uint8_t)0x01
```

Definition at line **408** of file **f\_nano\_crypto\_util.h**.

#### 5.3.2.2 MAX\_STR\_NANO\_CHAR

```
#define MAX_STR_NANO_CHAR (size_t)70
```

Defines a max size of Nano char (70 bytes)

Definition at line **129** of file **f\_nano\_crypto\_util.h**.

#### 5.3.2.3 NANO\_ENCRYPTED\_SEED\_FILE

```
#define NANO_ENCRYPTED_SEED_FILE "/spiffs/secure/nano.nse"
```

Path to non deterministic encrypted file with password.

File containing the SEED of the Nano wallets generated by TRNG (if available in your Hardware) or PRNG.  
Default name: "nano.nse"

Definition at line **171** of file **f\_nano\_crypto\_util.h**.

#### 5.3.2.4 NANO\_FILE\_WALLETS\_INFO

```
#define NANO_FILE_WALLETS_INFO "/spiffs/secure/walletsinfo.i"
```

Custom information file path about Nano SEED wallet stored in "walletsinfo.i".

Definition at line **189** of file **f\_nano\_crypto\_util.h**.

#### 5.3.2.5 NANO\_PASSWD\_MAX\_LEN

```
#define NANO_PASSWD_MAX_LEN (size_t)80
```

Password max length.

Definition at line **177** of file **f\_nano\_crypto\_util.h**.

### 5.3.2.6 NANO\_PREFIX

```
#define NANO_PREFIX "nano_"
```

Nano prefix.

Definition at line **141** of file **f\_nano\_crypto\_util.h**.

### 5.3.2.7 PUB\_KEY\_EXTENDED\_MAX\_LEN

```
#define PUB_KEY_EXTENDED_MAX_LEN (size_t)40
```

Max size of public key (extended)

Definition at line **135** of file **f\_nano\_crypto\_util.h**.

### 5.3.2.8 REP\_XRB

```
#define REP_XRB (uint8_t)0x4
```

Representative XRB flag.

Destination XRB flag.

Sender XRB flag.

### 5.3.2.9 SENDER\_XRB

```
#define SENDER_XRB (uint8_t)0x02
```

Definition at line **402** of file **f\_nano\_crypto\_util.h**.

### 5.3.2.10 STR\_NANO\_SZ

```
#define STR_NANO_SZ (size_t)66
```

String size of Nano encoded Base32 including NULL char.

Definition at line **183** of file **f\_nano\_crypto\_util.h**.

#### 5.3.2.11 `XRB_PREFIX`

```
#define XRB_PREFIX "xrb_"
```

XRB (old Raiblocks) prefix.

Definition at line **147** of file `f_nano_crypto_util.h`.

### 5.3.3 Typedef Documentation

#### 5.3.3.1 `F_FILE_INFO_ERR`

```
F_FILE_INFO_ERR
```

Typedef Error enumerator for info file functions.

#### 5.3.3.2 `f_nano_err`

```
f_nano_err
```

Error function enumerator.

See also

`f_nano_err_t` (p. ??)

#### 5.3.3.3 `f_uint128_t`

```
f_uint128_t
```

128 bit big number of Nano balance

Definition at line **201** of file `f_nano_crypto_util.h`.

#### 5.3.3.4 `f_write_seed_err`

```
typedef enum f_write_seed_err_t f_write_seed_err
```

#### 5.3.3.5 NANO\_PRIVATE\_KEY

NANO\_PRIVATE\_KEY

Size of Nano Private Key.

Definition at line 211 of file **f\_nano\_crypto\_util.h**.

#### 5.3.3.6 NANO\_PRIVATE\_KEY\_EXTENDED

NANO\_PRIVATE\_KEY\_EXTENDED

Size of Nano Private Key extended.

Definition at line 217 of file **f\_nano\_crypto\_util.h**.

#### 5.3.3.7 NANO\_PUBLIC\_KEY

NANO\_PUBLIC\_KEY

Size of Nano Public Key.

Definition at line 223 of file **f\_nano\_crypto\_util.h**.

#### 5.3.3.8 NANO\_PUBLIC\_KEY\_EXTENDED

NANO\_PUBLIC\_KEY\_EXTENDED

Size of Public Key Extended.

Definition at line 229 of file **f\_nano\_crypto\_util.h**.

#### 5.3.3.9 NANO\_SEED

NANO\_SEED

Size of Nano SEED.

Definition at line 195 of file **f\_nano\_crypto\_util.h**.

### 5.3.4 Enumeration Type Documentation

#### 5.3.4.1 f\_file\_info\_err\_t

enum **f\_file\_info\_err\_t**



## Enumerator

F_FILE_INFO_ERR_OK	SUCCESS.
F_FILE_INFO_ERR_CANT_OPEN_INFO_FILE	Can't open info file.
F_FILE_INFO_ERR_NANO_SEED_ENCRYPTED_FILE_NOT_FOUND	Encrypted file with Nano SEED not found.
F_FILE_INFO_ERR_CANT_DELETE_NANO_INFO_FILE	Can not delete Nano info file.
F_FILE_INFO_ERR_MALLOC	Fatal Error MALLOC.
F_FILE_INFO_ERR_CANT_READ_NANO_SEED_ENCRYPTED_FILE	Can not read encrypted Nano SEED in file.
F_FILE_INFO_ERR_CANT_READ_INFO_FILE	Can not read info file.
F_FILE_INFO_INVALID_HEADER_FILE	Invalid info file header.
F_FILE_INFO_ERR_INVALID_SHA256_INFO_FILE	Invalid SHA256 info file.
F_FILE_INFO_ERR_NANO_SEED_HASH_FAIL	Nano SEED hash failed.
F_FILE_INFO_ERR_NANO_INVALID_REPRESENTATIVE	Invalid representative.
F_FILE_INFO_ERR_NANO_INVALID_MAX_FEE_VALUE	Invalid max fee value.
F_FILE_INFO_ERR_OPEN_FOR_WRITE_INFO	Can not open info file for write.
F_FILE_INFO_ERR_EXISTING_FILE	Error File Exists.
F_FILE_INFO_ERR_CANT_WRITE_FILE_INFO	Can not write info file.

Definition at line 514 of file f\_nano\_crypto\_util.h.

## 5.3.4.2 f\_nano\_err\_t

```
enum f_nano_err_t
```

## Enumerator

NANO_ERR_OK	SUCCESS.
NANO_ERR_CANT_PARSE_BN_STR	Can not parse string big number.
NANO_ERR_MALLOC	Fatal ERROR MALLOC.
NANO_ERR_CANT_PARSE_FACTOR	Can not parse big number factor.
NANO_ERR_MPI_MULT	Error multiplication MPI.
NANO_ERR_CANT_PARSE_TO_BLK_TRANSFER	Can not parse to block transfer.
NANO_ERR_EMPTY_STR	Error empty string.
NANO_ERR_CANT_PARSE_VALUE	Can not parse value.
NANO_ERR_PARSE_MPI_TO_STR	Can not parse MPI to string.
NANO_ERR_CANT_COMPLETE_NULL_CHAR	Can not complete NULL char.
NANO_ERR_CANT_PARSE_TO_MPI	Can not parse to MPI.
NANO_ERR_INSUFICIENT_FUNDS	Insuficient funds.
NANO_ERR_SUB_MPI	Error subtract MPI.
NANO_ERR_ADD_MPI	Error add MPI.
NANO_ERR_NO_SENSE_VALUE_TO_SEND_NEGATIVE	Does not make sense send negativative balance.
NANO_ERR_NO_SENSE_VALUE_TO_SEND_ZERO	Does not make sense send empty value.
NANO_ERR_NO_SENSE_BALANCE_NEGATIVE	Does not make sense negative balance.
NANO_ERR_VAL_A_INVALID_MODE	Invalid A mode value.
NANO_ERR_CANT_PARSE_TO_TEMP_UINT128_T	Can not parse temporary memory to uint_128_t.
NANO_ERR_VAL_B_INVALID_MODE	Invalid A mode value.

## Enumerator

NANO_ERR_CANT_PARSE_RAW_A_TO_MPI	Can not parse raw A value to MPI.
NANO_ERR_CANT_PARSE_RAW_B_TO_MPI	Can not parse raw B value to MPI.
NANO_ERR_UNKNOWN_ADD_SUB_MODE	Unknown ADD/SUB mode.
NANO_ERR_INVALID_RES_OUTPUT	Invalid output result.

Definition at line **273** of file **f\_nano\_crypto\_util.h**.

## 5.3.4.3 f\_write\_seed\_err\_t

```
enum f_write_seed_err_t
```

## Enumerator

WRITE_ERR_OK	Error SUCCESS.
WRITE_ERR_NULL_PASSWORD	Error NULL password.
WRITE_ERR_EMPTY_STRING	Empty string.
WRITE_ERR_MALLOC	Error MALLOC.
WRITE_ERR_ENCRYPT_PRIV_KEY	Error encrypt private key.
WRITE_ERR_GEN_SUB_PRIV_KEY	Can not generate sub private key.
WRITE_ERR_GEN_MAIN_PRIV_KEY	Can not generate main private key.
WRITE_ERR_ENCRYPT_SUB_BLOCK	Can not encrypt sub block.
WRITE_ERR_UNKNOWN_OPTION	Unknown option.
WRITE_ERR_FILE_ALREADY_EXISTS	File already exists.
WRITE_ERR_CREATING_FILE	Can not create file.
WRITE_ERR_WRITING_FILE	Can not write file.

Definition at line **410** of file **f\_nano\_crypto\_util.h**.

## 5.3.5 Function Documentation

## 5.3.5.1 \_\_attribute\_\_()

```
struct f_nano_wallet_info_t __attribute__ (  
    (packed) )
```

#### 5.3.5.2 f\_bip39\_to\_nano\_seed()

```
int f_bip39_to_nano_seed (
    uint8_t * seed,
    char * str,
    char * dictionary )
```

Parse Nano Bip39 encoded string to raw Nano SEED given a dictionary file.

## Parameters

out	<i>seed</i>	Nano SEED
in	<i>str</i>	A encoded Bip39 string pointer
in	<i>dictionary</i>	A string pointer path to file

WARNING Sensitive data. Do not share any SEED or Bip39 encoded string !

## Return values

0	On Success, otherwise Error
---	-----------------------------

## See also

**f\_nano\_seed\_to\_bip39()** (p. ??)

## 5.3.5.3 f\_cloud\_crypto\_wallet\_nano\_create\_seed()

```
int f_cloud_crypto_wallet_nano_create_seed (
    size_t entropy,
    char * file_name,
    char * password )
```

Generates a new SEED and saves it to an non deterministic encrypted file.

*password* is mandatory

## Parameters

in	<i>entropy</i>	Entropy type. Entropy type are:  F_ENTROPY_TYPE_PARANOIC F_ENTROPY_TYPE_EXCELENT F_ENTROPY_TYPE_GOOD F_ENTROPY_TYPE_NOT_ENOUGH F_ENTROPY_TYPE_NOT_RECOMENDED
in	<i>file_name</i>	The file and path to be stored in your file system directory. It can be <i>NULL</i> . If you parse a <i>NULL</i> value then file will be stored in <i>NANO_ENCRYPTED_SEED_FILE</i> variable file system pointer.
in	<i>password</i>	Password of the encrypted file. It can NOT be <i>NULL</i> or EMPTY

## WARNING

**f\_cloud\_crypto\_wallet\_nano\_create\_seed()** (p. ??) does not verify your password. It is recommended to use a strong password like symbols, capital letters and numbers to keep your SEED safe and avoid brute force attacks.

You can use **f\_pass\_must\_have\_at\_least()** (p. ??) function to check passwords strenght

## Return values

0	On Success, otherwise Error
---	-----------------------------

## 5.3.5.4 f\_generate\_nano\_seed()

```
int f_generate_nano_seed (
    NANO_SEED seed,
    uint32_t entropy )
```

Generates a new SEED and stores it to *seed* pointer.

## Parameters

out	<i>seed</i>	SEED generated in system PRNG or TRNG
in	<i>entropy</i>	Entropy type. Entropy type are:  F_ENTROPY_TYPE_PARANOIC F_ENTROPY_TYPE_EXCELENT F_ENTROPY_TYPE_GOOD F_ENTROPY_TYPE_NOT_ENOUGH F_ENTROPY_TYPE_NOT_RECOMENDED

## Return values

0	On Success, otherwise Error
---	-----------------------------

## 5.3.5.5 f\_get\_nano\_file\_info()

```
F_FILE_INFO_ERR f_get_nano_file_info (
    F_NANO_WALLET_INFO * info )
```

Opens default file *walletsinfo.i* (if exists) containing information *F\_NANO\_WALLET\_INFO* structure and parsing to pointer *info* if success.

## Parameters

out	<i>info</i>	Pointer to buffer to be parsed struct from <i>\$PATH/walletsinfo.i</i> file.
-----	-------------	--

## Return values

<i>F_FILE_INFO_ERR_OK</i>	If Success, otherwise <i>F_FILE_INFO_ERR</i> enum type error
---------------------------	--

See also

**F\_FILE\_INFO\_ERR** (p. ??) enum type error for detailed error and **f\_nano\_wallet\_info\_t** (p. ??) for info type details

#### 5.3.5.6 f\_nano\_add\_sub()

```
f_nano_err f_nano_add_sub (
    void * res,
    void * valA,
    void * valB,
    uint32_t mode )
```

Add/Subtract two Nano balance values and stores value in *res*

##### Parameters

out	<i>res</i>	Result value $res = valA + valB$ or $res = valA - valB$
in	<i>valA</i>	Input balance A value
in	<i>valB</i>	Input balance B value
in	<i>mode</i>	Mode type: <ul style="list-style-type: none"> <li>• <i>F_NANO_ADD_A_B</i> <math>valA + valB</math></li> <li>• <i>F_NANO_SUB_A_B</i> <math>valA - valB</math></li> <li>• <i>F_NANO_A_RAW_128</i> if <i>balance</i> is big number raw buffer type</li> <li>• <i>F_NANO_A_RAW_STRING</i> if <i>balance</i> is big number raw string type</li> <li>• <i>F_NANO_A_REAL_STRING</i> if <i>balance</i> is real number string type</li> <li>• <i>F_NANO_B_RAW_128</i> if <i>value_to_send</i> is big number raw buffer type</li> <li>• <i>F_NANO_B_RAW_STRING</i> if <i>value_to_send</i> is big number raw string type</li> <li>• <i>F_NANO_B_REAL_STRING</i> if <i>value_to_send</i> is real number string type</li> </ul>

##### Return values

<i>NANO_ERR_OK</i>	If Success, otherwise <b>f_nano_err_t</b> enum type error
--------------------	---

See also

**f\_nano\_err\_t** (p. ??) for **f\_nano\_err** (p. ??) enum error type

#### 5.3.5.7 f\_nano\_key\_to\_str()

```
char * f_nano_key_to_str (
    char * out,
    unsigned char * key )
```

Parse a raw binary public key to string.

**Parameters**

out	<i>out</i>	Pointer to output string
in	<i>in</i>	Pointer to raw public key

**Returns**

A pointer to output string

**5.3.5.8 f\_nano\_parse\_raw\_str\_to\_raw128\_t()**

```
f_nano_err f_nano_parse_raw_str_to_raw128_t (
    uint8_t * res,
    const char * raw_str_value )
```

Parse a raw string balance to raw big number 128 bit.

**Parameters**

out	<i>res</i>	Binary raw balance
in	<i>raw_str_value</i>	Raw balance string

**Return values**

<i>NANO_ERR_OK</i>	If Success, otherwise f_nano_err_t enum type error
--------------------	--

**See also**

**f\_nano\_err\_t** (p. ??) for **f\_nano\_err** (p. ??) enum error type

**5.3.5.9 f\_nano\_parse\_real\_str\_to\_raw128\_t()**

```
f_nano_err f_nano_parse_real_str_to_raw128_t (
    uint8_t * res,
    const char * real_str_value )
```

Parse a real string balance to raw big number 128 bit.

**Parameters**

out	<i>res</i>	Binary raw balance
in	<i>real_str_value</i>	Real balance string



## Return values

<code>NANO_ERR_OK</code>	If Success, otherwise f_nano_err_t enum type error
--------------------------	--

## See also

`f_nano_err_t` (p. ??) for `f_nano_err` (p. ??) enum error type

## 5.3.5.10 f\_nano\_raw\_to\_string()

```
int f_nano_raw_to_string (
    char * str,
    size_t * olen,
    size_t str_sz,
    void * raw,
    int raw_type )
```

Converts Nano raw balance [string | f\_uint128\_t] to real string value.

## Parameters

out	<i>str</i>	Output real string value
out	<i>olen</i>	Size of output real string value. It can be NULL. If NULL output <i>str</i> will have a NULL char at the end.
in	<i>str_sz</i>	Size of <i>str</i> buffer
in	<i>raw</i>	Raw balance.
in	<i>raw_type</i>	Raw balance type: <ul style="list-style-type: none"> <li>• F_RAW_TO_STR_UINT128 for raw <b>f_uint128_t</b> balance</li> <li>• F_RAW_TO_STR_STRING for raw <b>char</b> balance</li> </ul>

## Return values

<code>0</code>	On Success, otherwise Error
----------------	-----------------------------

## See also

`f_nano_valid_nano_str_value()` (p. ??)

## 5.3.5.11 f\_nano\_seed\_to\_bip39()

```
int f_nano_seed_to_bip39 (
    char * buf,
```

```

size_t buf_sz,
size_t * out_buf_len,
NANO_SEED seed,
char * dictionary_file )

```

Parse Nano SEED to Bip39 encoding given a dictionary file.

#### Parameters

out	<i>buf</i>	Output string containing encoded Bip39 SEED
in	<i>buf_sz</i>	Size of memory of buf pointer
out	<i>out_buf_len</i>	If <i>out_buf_len</i> is NOT NULL then <i>out_buf_len</i> returns the size of string encoded Bip39 and <i>out</i> with non NULL char. If <i>out_buf_len</i> is NULL then <i>out</i> has a string encoded Bip39 with a NULL char.
in	<i>seed</i>	Nano SEED
in	<i>dictionary_file</i>	Path to dictionary file

WARNING Sensitive data. Do not share any SEED or Bip39 encoded string !

#### Return values

0	On Success, otherwise Error
---	-----------------------------

#### See also

**f\_bip39\_to\_nano\_seed()** (p. ??)

#### 5.3.5.12 f\_nano\_sign\_block()

```

int f_nano_sign_block (
    F_BLOCK_TRANSFER * user_block,
    F_BLOCK_TRANSFER * fee_block,
    NANO_PRIVATE_KEY_EXTENDED private_key )

```

Signs *user\_block* and worker *fee\_block* given a private key *private\_key*

#### Parameters

in, out	<i>user_block</i>	User block to be signed with a private key <i>private_key</i>
in, out	<i>fee_block</i>	Fee block to be signed with a private key <i>private_key</i> . Can be NULL if worker does not require fee
in	<i>private_key</i>	Private key to sign block(s)

#### Return values

0	If Success, otherwise error
---	-----------------------------

See also

**f\_nano\_transaction\_to\_JSON()** (p. ??)

#### 5.3.5.13 f\_nano\_transaction\_to\_JSON()

```
int f_nano_transaction_to_JSON (
    char * str,
    size_t str_len,
    size_t * str_out,
    NANO_PRIVATE_KEY_EXTENDED private_key,
    F_BLOCK_TRANSFER * block_transfer )
```

Sign a block pointed in *block\_transfer* with a given *private\_key* and stores signed block to *block\_transfer* and parse to JSON Nano RPC.

##### Parameters

out	<i>str</i>	A string pointer to store JSON Nano RPC
in	<i>str_len</i>	Size of buffer in <i>str</i> pointer
out	<i>str_out</i>	Size of JSON string. <i>str_out</i> can be NULL
in	<i>private_key</i>	Private key to sign the block <i>block_transfer</i>
in, out	<i>block_transfer</i>	Nano block containing raw data to be stored in Nano Blockchain

WARNING Sensitive data. Do not share any PRIVATE KEY

##### Return values

0	On Success, otherwise Error
---	-----------------------------

#### 5.3.5.14 f\_nano\_valid\_nano\_str\_value()

```
int f_nano_valid_nano_str_value (
    const char * str )
```

Check if a real string or raw string are valid Nano balance.

##### Parameters

in	<i>str</i>	Value to be checked
----	------------	---------------------

##### Return values

0	If valid, otherwise is invalid
---	--------------------------------

See also

**f\_nano\_raw\_to\_string()** (p. ??)

### 5.3.5.15 f\_nano\_value\_compare\_value()

```
f_nano_err f_nano_value_compare_value (
    void * valA,
    void * valB,
    uint32_t * mode_compare )
```

Compare two Nano balance.

#### Parameters

in	<i>valA</i>	Nano balance value A
in	<i>valB</i>	Nano balance value B
in, out	<i>mode_compare</i>	<p>Input mode and output result</p> <p>Input mode:</p> <ul style="list-style-type: none"> <li>• <i>F_NANO_A_RAW_128</i> if <i>valA</i> is big number raw buffer type</li> <li>• <i>F_NANO_A_RAW_STRING</i> if <i>valA</i> is big number raw string type</li> <li>• <i>F_NANO_A_REAL_STRING</i> if <i>valA</i> is real number string type</li> <li>• <i>F_NANO_B_RAW_128</i> if <i>valB</i> is big number raw buffer type</li> <li>• <i>F_NANO_B_RAW_STRING</i> if <i>valB</i> is big number raw string type</li> <li>• <i>F_NANO_B_REAL_STRING</i> if <i>valB</i> is real number string type</li> </ul> <p>Output type:</p> <ul style="list-style-type: none"> <li>• <i>F_NANO_COMPARE_EQ</i> If <i>valA</i> is greater than <i>valB</i></li> <li>• <i>F_NANO_COMPARE_LT</i> if <i>valA</i> is lesser than <i>valB</i></li> <li>• <i>F_NANO_COMPARE_LEQ</i> if <i>valA</i> is lesser or equal than <i>valB</i></li> <li>• <i>F_NANO_COMPARE_GT</i> if <i>valA</i> is greater than <i>valB</i></li> <li>• <i>F_NANO_COMPARE_GEQ</i> If <i>valA</i> is greater or equal than <i>valB</i></li> </ul>

#### Return values

<i>NANO_ERR_OK</i>	If Success, otherwise <b>f_nano_err_t</b> enum type error
--------------------	---

See also

**f\_nano\_err\_t** (p. ??) for **f\_nano\_err** (p. ??) enum error type

## 5.3.5.16 f\_nano\_verify\_nano\_funds()

```

f_nano_err f_nano_verify_nano_funds (
    void * balance,
    void * value_to_send,
    void * fee,
    uint32_t mode )

```

Check if Nano balance has sufficient funds.

## Parameters

in	<i>balance</i>	Nano balance
in	<i>value_to_send</i>	Value to send
in	<i>fee</i>	Fee value (it can be NULL)
in	<i>mode</i>	Value type mode <ul style="list-style-type: none"> <li>• <i>F_NANO_A_RAW_128</i> if <i>balance</i> is big number raw buffer type</li> <li>• <i>F_NANO_A_RAW_STRING</i> if <i>balance</i> is big number raw string type</li> <li>• <i>F_NANO_A_REAL_STRING</i> if <i>balance</i> is real number string type</li> <li>• <i>F_NANO_B_RAW_128</i> if <i>value_to_send</i> is big number raw buffer type</li> <li>• <i>F_NANO_B_RAW_STRING</i> if <i>value_to_send</i> is big number raw string type</li> <li>• <i>F_NANO_B_REAL_STRING</i> if <i>value_to_send</i> is real number string type</li> <li>• <i>F_NANO_C_RAW_128</i> if <i>fee</i> is big number raw buffer type (can be omitted if <i>fee</i> is NULL)</li> <li>• <i>F_NANO_C_RAW_STRING</i> if <i>fee</i> is big number raw string type (can be omitted if <i>fee</i> is NULL)</li> <li>• <i>F_NANO_C_REAL_STRING</i> if <i>fee</i> is real number string type (can be omitted if <i>fee</i> is NULL)</li> </ul>

## Return values

<i>NANO_ERR_OK</i>	If Success, otherwise f_nano_err_t enum type error
--------------------	--

## See also

**f\_nano\_err\_t** (p. ??) for **f\_nano\_err** (p. ??) enum error type

## 5.3.5.17 f\_parse\_nano\_seed\_and\_bip39\_to\_JSON()

```

int f_parse_nano_seed_and_bip39_to_JSON (
    char * dest,
    size_t dest_sz,
    size_t * olen,
    void * source_data,

```

```
int source,
const char * password )
```

Parse Nano SEED and Bip39 to JSON given a encrypted data in memory or encrypted data in file or unencrypted seed in memory.

#### Parameters

out	<i>dest</i>	Destination JSON string pointer
in	<i>dest_sz</i>	Buffer size of <i>dest</i> pointer
out	<i>olen</i>	Size of the output JSON string. If NULL string JSON returns a NULL char at the end of string otherwise it will return the size of the string is stored into <i>olen</i> variable without NULL string in <i>dest</i>
in	<i>source_data</i>	Input data source (encrypted file   encrypted data in memory   unencrypted seed in memory)
in	<i>source</i>	Source data type: <ul style="list-style-type: none"> <li>• PARSE_JSON_READ_SEED_GENERIC: If seed are in memory pointed in <i>source_data</i>. Password is ignored. Can be NULL.</li> <li>• READ_SEED_FROM_STREAM: Read encrypted data from stream pointed in <i>source_data</i>. Password is required.</li> <li>• READ_SEED_FROM_FILE: Read encrypted data stored in a file where <i>source_data</i> is path to file. Password is required.</li> </ul>
in	<i>password</i>	Required for READ_SEED_FROM_STREAM and READ_SEED_FROM_FILE sources

WARNING Sensitive data. Do not share any SEED or Bip39 encoded string !

#### Return values

0	On Success, otherwise Error
---	-----------------------------

#### See also

**f\_read\_seed()** (p. ??)

#### 5.3.5.18 f\_read\_seed()

```
int f_read_seed (
    uint8_t * seed,
    const char * passwd,
    void * source_data,
    int force_read,
    int source )
```

Extracts a Nano SEED from encrypted stream in memory or in a file.

## Parameters

out	<i>seed</i>	Output Nano SEED
in	<i>passwd</i>	Password (always required)
in	<i>source_data</i>	Encrypted source data from memory or path pointed in <i>source_data</i>
in	<i>force_read</i>	If non zero value then forces reading from a corrupted file. This param is ignored when reading <i>source_data</i> from memory
in	<i>source</i>	Source data type: <ul style="list-style-type: none"> <li>• <b>READ_SEED_FROM_STREAM</b>: Read encrypted data from stream pointed in <i>source_data</i>. Password is required.</li> <li>• <b>READ_SEED_FROM_FILE</b>: Read encrypted data stored in a file where <i>source_data</i> is path to file. Password is required.</li> </ul>

WARNING Sensitive data. Do not share any SEED !

## Return values

0	On Success, otherwise Error
---	-----------------------------

## See also

**f\_parse\_nano\_seed\_and\_bip39\_to\_JSON()** (p. ??)

## 5.3.5.19 f\_seed\_to\_nano\_wallet()

```
int f_seed_to_nano_wallet (
    NANO_PRIVATE_KEY private_key,
    NANO_PUBLIC_KEY public_key,
    NANO_SEED seed,
    uint32_t wallet_number )
```

Extracts one key pair from Nano SEED given a wallet number.

## Parameters

out	<i>private_key</i>	Private key of the <i>wallet_number</i> from given <i>seed</i>
out	<i>public_key</i>	Public key of the <i>wallet_number</i> from given <i>seed</i>
in, out	<i>seed</i>	Nano SEED
in	<i>wallet_number</i>	Wallet number of key pair to be extracted from Nano SEED

## WARNING 1:

- Seed must be read from memory
- Seed is destroyed when extracting public and private keys

## WARNING 2:

- Never expose SEED and private key. This function destroys seed and any data after execution and finally parse public and private keys to output.

## Return values

0	On Success, otherwise Error
---	-----------------------------

## 5.3.5.20 f\_set\_nano\_file\_info()

```
F_FILE_INFO_ERR f_set_nano_file_info (
    F_NANO_WALLET_INFO * info,
    int overwrite_existing_file )
```

Saves wallet information stored at buffer struct *info* to file *walletsinfo.i*

## Parameters

in	<i>info</i>	Pointer to data to be saved at <i>\$PATH/walletsinfo.i</i> file.
in	<i>overwrite_existing_file</i>	If non zero then overwrites file <i>\$PATH/walletsinfo.i</i>

## Return values

<i>F_FILE_INFO_ERR_OK</i>	If Success, otherwise <i>F_FILE_INFO_ERR</i> enum type error
---------------------------	--

## See also

**F\_FILE\_INFO\_ERR** (p. ??) enum type error for detailed error and **f\_nano\_wallet\_info\_t** (p. ??) for info type details

## 5.3.5.21 is\_nano\_prefix()

```
int is_nano_prefix (
    const char * nano_wallet,
    const char * prefix )
```

Checks *prefix* in *nano\_wallet*

## Parameters

in	<i>nano_wallet</i>	Base32 Nano wallet encoded string
in	<i>prefix</i>	Prefix type <ul style="list-style-type: none"> <li>• NANO_PREFIX for nano_</li> <li>• XRB_PREFIX for xrb_</li> </ul>



## Return values

1	If <i>prefix</i> in <i>nano_wallet</i> , otherwise 0
---	--

## 5.3.5.22 is\_null\_hash()

```
int is_null_hash (
    uint8_t * hash )
```

Check if 32 bytes hash is filled with zeroes.

## Parameters

in	<i>hash</i>	32 bytes binary <i>hash</i>
----	-------------	-----------------------------

## Return values

1	If zero filled buffer, otherwise 0
---	------------------------------------

## 5.3.5.23 nano\_base\_32\_2\_hex()

```
int nano_base_32_2_hex (
    uint8_t * res,
    char * str_wallet )
```

Parse Nano Base32 wallet string to public key binary.

## Parameters

out	<i>res</i>	Output raw binary public key
in	<i>str_wallet</i>	Valid Base32 encoded Nano string to be parsed

## Return values

0	On Success, otherwise Error
---	-----------------------------

## See also

**pk\_to\_wallet()** (p. ??)

5.3.5.24 `pk_to_wallet()`

```
int pk_to_wallet (
    char * out,
    char * prefix,
    NANO_PUBLIC_KEY_EXTENDED pubkey_extended )
```

Parse a Nano public key to Base32 Nano wallet string.

## Parameters

out	<i>out</i>	Output string containing the wallet
in	<i>prefix</i>	Nano prefix.  <i>NANO_PREFIX</i> for nano_ <i>XRB_PREFIX</i> for xrb_
in, out	<i>pubkey_extended</i>	Public key to be parsed to string

WARNING: *pubkey\_extended* is destroyed when parsing to Nano base32 encoding

## Return values

0	On Success, otherwise Error
---	-----------------------------

## See also

**nano\_base\_32\_2\_hex()** (p. ??)

5.3.5.25 `valid_nano_wallet()`

```
int valid_nano_wallet (
    const char * wallet )
```

Check if a string containing a Base32 Nano wallet is valid.

## Parameters

in	<i>wallet</i>	Base32 Nano wallet encoded string
----	---------------	-----------------------------------

## Return values

0	If valid wallet otherwise is invalid
---	--------------------------------------

**5.3.5.26 valid\_raw\_balance()**

```
int valid_raw_balance (
    const char * balance )
```

Checks if a string buffer pointed in *balance* is a valid raw balance.

**Parameters**

in	<i>balance</i>	Pointer containing a string buffer
----	----------------	------------------------------------

**Return values**

0	On Success, otherwise Error
---	-----------------------------

**5.3.6 Variable Documentation****5.3.6.1 account**

```
uint8_t account[32]
```

Account in raw binary data.

Definition at line **233** of file **f\_nano\_crypto\_util.h**.

**5.3.6.2 balance**

```
f_uint128_t balance
```

Big number 128 bit raw balance.

See also

**f\_uint128\_t** (p. ??)

Definition at line **241** of file **f\_nano\_crypto\_util.h**.

**5.3.6.3 body**

```
F_NANO_WALLET_INFO_BODY body
```

Body of the file info.

Definition at line **241** of file **f\_nano\_crypto\_util.h**.

#### 5.3.6.4 desc

```
char desc[F_NANO_DESC_SZ]
```

Description.

Definition at line **235** of file **f\_nano\_crypto\_util.h**.

#### 5.3.6.5 description

```
uint8_t description[F_DESC_SZ]
```

File description.

Definition at line **235** of file **f\_nano\_crypto\_util.h**.

#### 5.3.6.6 file\_info\_integrity

```
uint8_t file_info_integrity[32]
```

File info integrity of the body block.

Definition at line **239** of file **f\_nano\_crypto\_util.h**.

#### 5.3.6.7 hash\_sk\_unencrypted

```
uint8_t hash_sk_unencrypted[32]
```

hash of Nano SEED when unencrypted

Definition at line **237** of file **f\_nano\_crypto\_util.h**.

#### 5.3.6.8 header

```
uint8_t header[sizeof(F_NANO_WALLET_INFO_MAGIC)]
```

Header magic.

Definition at line **231** of file **f\_nano\_crypto\_util.h**.

#### 5.3.6.9 iv

```
uint8_t iv
```

Initial sub vector.

Initial vector of first encryption layer.

Definition at line **233** of file **f\_nano\_crypto\_util.h**.

#### 5.3.6.10 last\_used\_wallet\_number

```
uint32_t last_used_wallet_number
```

Last used wallet number.

Definition at line **233** of file **f\_nano\_crypto\_util.h**.

#### 5.3.6.11 link

```
uint8_t link[32]
```

link or destination account

Definition at line **243** of file **f\_nano\_crypto\_util.h**.

#### 5.3.6.12 max\_fee

```
char max_fee[F_RAW_STR_MAX_SZ]
```

Custom preferred max fee of Proof of Work.

Definition at line **237** of file **f\_nano\_crypto\_util.h**.

#### 5.3.6.13 nano\_hdr

```
uint8_t nano_hdr[sizeof(NANO_WALLET_MAGIC)]
```

Header of the file.

Definition at line **231** of file **f\_nano\_crypto\_util.h**.

#### 5.3.6.14 nanoseed\_hash

```
uint8_t nanoseed_hash[32]
```

Nano SEED hash file.

Definition at line **237** of file **f\_nano\_crypto\_util.h**.

#### 5.3.6.15 preamble

```
uint8_t preamble[32]
```

Block preamble.

Definition at line **231** of file **f\_nano\_crypto\_util.h**.

#### 5.3.6.16 prefixes

```
uint8_t prefixes
```

Internal use for this API.

Definition at line **247** of file **f\_nano\_crypto\_util.h**.

#### 5.3.6.17 previous

```
uint8_t previous[32]
```

Previous block.

Definition at line **235** of file **f\_nano\_crypto\_util.h**.

#### 5.3.6.18 representative

```
uint8_t representative[32]
```

Representative for current account.

Definition at line **237** of file **f\_nano\_crypto\_util.h**.

#### 5.3.6.19 `reserved`

```
uint8_t reserved
```

Reserved (not used)

Reserved.

Definition at line **235** of file `f_nano_crypto_util.h`.

#### 5.3.6.20 `salt`

```
uint8_t salt[32]
```

Salt of the first encryption layer.

Definition at line **237** of file `f_nano_crypto_util.h`.

#### 5.3.6.21 `seed_block`

```
F_ENCRYPTED_BLOCK seed_block
```

Second encrypted block for Nano SEED.

Definition at line **241** of file `f_nano_crypto_util.h`.

#### 5.3.6.22 `signature`

```
uint8_t signature[64]
```

Signature of the block.

Definition at line **245** of file `f_nano_crypto_util.h`.

#### 5.3.6.23 `sk_encrypted`

```
uint8_t sk_encrypted[32]
```

Secret.

SEED encrypted (second layer)

Definition at line **239** of file `f_nano_crypto_util.h`.

#### 5.3.6.24 sub\_salt

```
uint8_t sub_salt[32]
```

Salt of the sub block to be stored.

Definition at line 231 of file **f\_nano\_crypto\_util.h**.

#### 5.3.6.25 ver

```
uint32_t ver
```

Version of the file.

Definition at line 233 of file **f\_nano\_crypto\_util.h**.

#### 5.3.6.26 version

```
uint16_t version
```

Version.

Definition at line 233 of file **f\_nano\_crypto\_util.h**.

#### 5.3.6.27 wallet\_prefix

```
uint8_t wallet_prefix
```

Wallet prefix: 0 for NANO; 1 for XRB.

Definition at line 231 of file **f\_nano\_crypto\_util.h**.

#### 5.3.6.28 wallet\_representative

```
char wallet_representative[ MAX_STR_NANO_CHAR]
```

Wallet representative.

Definition at line 235 of file **f\_nano\_crypto\_util.h**.



## 5.3.6.29 work

uint64\_t work

Internal use for this API.

Definition at line 249 of file f\_nano\_crypto\_util.h.

## 5.4 f\_nano\_crypto\_util.h

```

00001  /*
00002     AUTHOR: Fábio Pereira da Silva
00003     YEAR: 2019-20
00004     LICENSE: MIT
00005     EMAIL: fabioegel@gmail.com or fabioegel@protonmail.com
00006  */
00007
00008  #include <stdint.h>
00009  #include "f_util.h"
00010
00011  #ifndef F_DOC_SKIP
00012
00013     #ifdef F_XTENZA
00014
00015         #ifndef F_ESP32
00016             #define F_ESP32
00017         #endif
00018
00019         #include "esp_system.h"
00020
00021     #endif
00022
00023     #include "sodium/crypto_generichash.h"
00024     #include "sodium/crypto_sign.h"
00025     #include "sodium.h"
00026
00027     #ifdef F_ESP32
00028
00029         #include "sodium/private/curve25519_ref10.h"
00030
00031     #else
00032
00033         #include "sodium/private/ed25519_ref10.h"
00034
00035         #define ge_p3 ge25519_p3
00036         #define sc_reduce sc25519_reduce
00037         #define sc_muladd sc25519_muladd
00038         #define ge_scalarmult_base ge25519_scalarmult_base
00039         #define ge_p3_tobytes ge25519_p3_tobytes
00040
00041     #endif
00042
00043 #endif
00044
00121 #ifdef __cplusplus
00122 extern "C" {
00123 #endif
00124
00129 #define MAX_STR_NANO_CHAR (size_t)70 //5+56+8+1
00130
00135 #define PUB_KEY_EXTENDED_MAX_LEN (size_t)40
00136
00141 #define NANO_PREFIX "nano_"
00142
00147 #define XRB_PREFIX "xrb_"
00148
00149 #ifdef F_ESP32
00150
00155 #define BIP39_DICTIONARY "/spiffs/dictionary.dic"
00156 #else
00157
00158     #ifndef F_DOC_SKIP
00159         #define BIP39_DICTIONARY "/spiffs/dictionary.dic"
00160         #define BIP39_DICTIONARY_SAMPLE "../dictionary.dic"
00161     #endif
00162
00163 #endif
00164

```

```

00171 #define NANO_ENCRYPTED_SEED_FILE "/spiffs/secure/nano.nse"
00172
00177 #define NANO_PASSWD_MAX_LEN (size_t)80
00178
00183 #define STR_NANO_SZ (size_t)66// 65+1 Null included
00184
00189 #define NANO_FILE_WALLETS_INFO "/spiffs/secure/walletsinfo.i"
00190
00195 typedef uint8_t NANO_SEED[crypto_sign_SEEDBYTES];
00196
00201 typedef uint8_t f_uint128_t[16];
00202
00203 #ifndef F_DOC_SKIP
00204 #define EXPORT_KEY_TO_CHAR_SZ (size_t)sizeof(NANO_SEED)+1
00205 #endif
00206
00211 typedef uint8_t NANO_PRIVATE_KEY[sizeof(NANO_SEED)];
00212
00217 typedef uint8_t NANO_PRIVATE_KEY_EXTENDED[crypto_sign_ed25519_SECRETKEYBYTES];
00218
00223 typedef uint8_t NANO_PUBLIC_KEY[crypto_sign_ed25519_PUBLICKEYBYTES];
00224
00229 typedef uint8_t NANO_PUBLIC_KEY_EXTENDED[PUB_KEY_EXTENDED_MAX_LEN];
00230
00239 typedef struct f_block_transfer_t {
00241     uint8_t preamble[32];
00243     uint8_t account[32];
00245     uint8_t previous[32];
00247     uint8_t representative[32];
00251     f_uint128_t balance;
00253     uint8_t link[32];
00255     uint8_t signature[64];
00257     uint8_t prefixes;
00259     uint64_t work;
00260 } __attribute__((packed)) F_BLOCK_TRANSFER;
00261
00262 #ifndef F_DOC_SKIP
00263 #define F_BLOCK_TRANSFER_SIGNABLE_SZ
00264     (size_t)(sizeof(F_BLOCK_TRANSFER)-64-sizeof(uint64_t)-sizeof(uint8_t))
00265 #endif
00266
00273 typedef enum f_nano_err_t {
00275     NANO_ERR_OK=0,
00277     NANO_ERR_CANT_PARSE_BN_STR=5151,
00279     NANO_ERR_MALLOC,
00281     NANO_ERR_CANT_PARSE_FACTOR,
00283     NANO_ERR_MPI_MULT,
00285     NANO_ERR_CANT_PARSE_TO_BLK_TRANSFER,
00287     NANO_ERR_EMPTY_STR,
00289     NANO_ERR_CANT_PARSE_VALUE,
00291     NANO_ERR_PARSE_MPI_TO_STR,
00293     NANO_ERR_CANT_COMPLETE_NULL_CHAR,
00295     NANO_ERR_CANT_PARSE_TO_MPI,
00297     NANO_ERR_INSUFFICIENT_FUNDS,
00299     NANO_ERR_SUB_MPI,
00301     NANO_ERR_ADD_MPI,
00303     NANO_ERR_NO_SENSE_VALUE_TO_SEND_NEGATIVE,
00305     NANO_ERR_NO_SENSE_VALUE_TO_SEND_ZERO,
00307     NANO_ERR_NO_SENSE_BALANCE_NEGATIVE,
00309     NANO_ERR_VAL_A_INVALID_MODE,
00311     NANO_ERR_CANT_PARSE_TO_TEMP_UINT128_T,
00313     NANO_ERR_VAL_B_INVALID_MODE,
00315     NANO_ERR_CANT_PARSE_RAW_A_TO_MPI,
00317     NANO_ERR_CANT_PARSE_RAW_B_TO_MPI,
00319     NANO_ERR_UNKNOWN_ADD_SUB_MODE,
00321     NANO_ERR_INVALID_RES_OUTPUT
00322 } f_nano_err;
00323
00324 #ifndef F_DOC_SKIP
00325
00326 #define READ_SEED_FROM_STREAM (int)1
00327 #define READ_SEED_FROM_FILE (int)2
00328 #define WRITE_SEED_TO_STREAM (int)4
00329 #define WRITE_SEED_TO_FILE (int)8
00330 #define PARSE_JSON_READ_SEED_GENERIC (int)16
00331 #define F_STREAM_DATA_FILE_VERSION (uint32_t)((1<<16)|0)
00332
00333 #endif
00334
00342 typedef struct f_nano_encrypted_wallet_t {
00344     uint8_t sub_salt[32];
00346     uint8_t iv[16];
00348     uint8_t reserved[16];
00350     uint8_t hash_sk_unencrypted[32];
00352     uint8_t sk_encrypted[32];
00353 } __attribute__((packed)) F_ENCRYPTED_BLOCK;
00354

```

```

00355 #ifndef F_DOC_SKIP
00356
00357 static const uint8_t NANO_WALLET_MAGIC[] = {'_', 'n', 'a', 'n', 'o', 'w', 'a', 'l', 'l', 'e', 't', 'f',
'i', 'l', 'e', '_'};
00358 #define F_NANO_FILE_DESC "NANO Seed Encrypted file/stream. Keep it safe and backup it. This file is
protected by password. BUY BITCOIN and NANO !!!"
00359 #define F_DESC_SZ (size_t) (160-sizeof(uint32_t))
00360
00361 #endif
00362
00370 typedef struct f_nano_crypto_wallet_t {
00372     uint8_t nano_hdr[sizeof(NANO_WALLET_MAGIC)];
00374     uint32_t ver;
00376     uint8_t description[F_DESC_SZ];
00378     uint8_t salt[32];
00380     uint8_t iv[16];
00382     F_ENCRYPTED_BLOCK seed_block;
00383 } __attribute__((packed)) F_NANO_CRYPTOWALLET;
00384
00385 #ifndef F_DOC_SKIP
00386
00387 _Static_assert((sizeof(F_NANO_CRYPTOWALLET)&0x1F)==0, "Error 1");
00388 _Static_assert((sizeof(F_ENCRYPTED_BLOCK)&0x1F)==0, "Error 2");
00389
00390 #endif
00391
00396 #define REP_XRB (uint8_t)0x4
00397
00402 #define SENDER_XRB (uint8_t)0x02
00403
00408 #define DEST_XRB (uint8_t)0x01
00409
00410 typedef enum f_write_seed_err_t {
00412     WRITE_ERR_OK=0,
00414     WRITE_ERR_NULL_PASSWORD=7180,
00416     WRITE_ERR_EMPTY_STRING,
00418     WRITE_ERR_MALLOC,
00420     WRITE_ERR_ENCRYPT_PRIV_KEY,
00422     WRITE_ERR_GEN_SUB_PRIV_KEY,
00424     WRITE_ERR_GEN_MAIN_PRIV_KEY,
00426     WRITE_ERR_ENCRYPT_SUB_BLOCK,
00428     WRITE_ERR_UNKNOWN_OPTION,
00430     WRITE_ERR_FILE_ALREADY_EXISTS,
00432     WRITE_ERR_CREATING_FILE,
00434     WRITE_ERR_WRITING_FILE
00435 } f_write_seed_err;
00436
00437 #ifndef F_DOC_SKIP
00438
00439 #define F_RAW_TO_STR_UINT128 (int)1
00440 #define F_RAW_TO_STR_STRING (int)2
00441 #define F_RAW_STR_MAX_SZ (size_t)41 // 39 + '\0' + '.' -> 39 = log10(2^128)
00442 #define F_MAX_STR_RAW_BALANCE_MAX (size_t)40 //39+'0'
00443 #define F_NANO_EMPTY_BALANCE "0.0"
00444
00445 #endif
00446
00454 typedef struct f_nano_wallet_info_bdy_t {
00456     uint8_t wallet_prefix; // 0 for NANO; 1 for XRB
00458     uint32_t last_used_wallet_number;
00460     char wallet_representative[MAX_STR_NANO_CHAR];
00462     char max_fee[F_RAW_STR_MAX_SZ];
00464     uint8_t reserved[44];
00465 } __attribute__((packed)) F_NANO_WALLET_INFO_BODY;
00466
00467 #ifndef F_DOC_SKIP
00468
00469 _Static_assert((sizeof(F_NANO_WALLET_INFO_BODY)&0x1F)==0, "Error F_NANO_WALLET_INFO_BODY is not byte
aligned");
00470
00471 #define F_NANO_WALLET_INFO_DESC "Nano file descriptor used for fast custom access. BUY BITCOIN AND NANO."
00472 #define F_NANO_WALLET_INFO_VERSION (uint16_t)((1<8)|1)
00473 static const uint8_t F_NANO_WALLET_INFO_MAGIC[] = {'_', 'n', 'a', 'n', 'o', 'w', 'a', 'l', 'l', 'e', 't',
'i', 'l', 'e', '_'};
00474
00475 #define F_NANO_DESC_SZ (size_t)78
00476
00477 #endif
00478
00486 typedef struct f_nano_wallet_info_t {
00488     uint8_t header[sizeof(F_NANO_WALLET_INFO_MAGIC)];
00490     uint16_t version;
00492     char desc[F_NANO_DESC_SZ];
00494     uint8_t nanoseed_hash[32];
00496     uint8_t file_info_integrity[32];
00498     F_NANO_WALLET_INFO_BODY body;
00499 } __attribute__((packed)) F_NANO_WALLET_INFO;

```

```

00500
00501 #ifndef F_DOC_SKIP
00502
00503 _Static_assert((sizeof(F_NANO_WALLET_INFO)&0x1F)==0, "Error F_NANO_WALLET_INFO is not byte aligned");
00504
00505 #endif
00506
00514 typedef enum f_file_info_err_t {
00516     F_FILE_INFO_ERR_OK=0,
00518     F_FILE_INFO_ERR_CANT_OPEN_INFO_FILE=7001,
00520     F_FILE_INFO_ERR_NANO_SEED_ENCRYPTED_FILE_NOT_FOUND,
00522     F_FILE_INFO_ERR_CANT_DELETE_NANO_INFO_FILE,
00524     F_FILE_INFO_ERR_MALLOC,
00526     F_FILE_INFO_ERR_CANT_READ_NANO_SEED_ENCRYPTED_FILE,
00528     F_FILE_INFO_ERR_CANT_READ_INFO_FILE,
00530     F_FILE_INFO_INVALID_HEADER_FILE,
00532     F_FILE_INFO_ERR_INVALID_SHA256_INFO_FILE,
00534     F_FILE_INFO_ERR_NANO_SEED_HASH_FAIL,
00536     F_FILE_INFO_ERR_NANO_INVALID_REPRESENTATIVE,
00538     F_FILE_INFO_ERR_NANO_INVALID_MAX_FEE_VALUE,
00540     F_FILE_INFO_ERR_OPEN_FOR_WRITE_INFO,
00542     F_FILE_INFO_ERR_EXISTING_FILE,
00544     F_FILE_INFO_ERR_CANT_WRITE_FILE_INFO
00545 } F_FILE_INFO_ERR;
00546
00547 #ifndef F_DOC_SKIP
00548
00549 #define F_NANO_ADD_A_B (uint32_t)(1<<0)
00550 #define F_NANO_SUB_A_B (uint32_t)(1<<1)
00551 #define F_NANO_A_RAW_128 (uint32_t)(1<<2)
00552 #define F_NANO_A_RAW_STRING (uint32_t)(1<<3)
00553 #define F_NANO_A_REAL_STRING (uint32_t)(1<<4)
00554 #define F_NANO_B_RAW_128 (uint32_t)(1<<5)
00555 #define F_NANO_B_RAW_STRING (uint32_t)(1<<6)
00556 #define F_NANO_B_REAL_STRING (uint32_t)(1<<7)
00557 #define F_NANO_RES_RAW_128 (uint32_t)(1<<8)
00558 #define F_NANO_RES_RAW_STRING (uint32_t)(1<<9)
00559 #define F_NANO_RES_REAL_STRING (uint32_t)(1<<10)
00560 #define F_NANO_C_RAW_128 (uint32_t)(F_NANO_B_RAW_128<<16)
00561 #define F_NANO_C_RAW_STRING (uint32_t)(F_NANO_B_RAW_STRING<<16)
00562 #define F_NANO_C_REAL_STRING (uint32_t)(F_NANO_B_REAL_STRING<<16)
00563
00564 #define F_NANO_COMPARE_EQ (uint32_t)(1<<16) //Equal
00565 #define F_NANO_COMPARE_LT (uint32_t)(1<<17) // Lesser than
00566 #define F_NANO_COMPARE_LEQ (F_NANO_COMPARE_LT|F_NANO_COMPARE_EQ) // Less or equal
00567 #define F_NANO_COMPARE_GT (uint32_t)(1<<18) // Greater
00568 #define F_NANO_COMPARE_GEQ (F_NANO_COMPARE_GT|F_NANO_COMPARE_EQ) // Greater or equal
00569 #define DEFAULT_MAX_FEE "0.001"
00570
00571 #endif
00572
00595 int f_cloud_crypto_wallet_nano_create_seed(size_t, char *, char *);
00596
00609 int f_generate_nano_seed(NANO_SEED, uint32_t);
00610
00625 int pk_to_wallet(char *, char *, NANO_PUBLIC_KEY_EXTENDED);
00626
00644 int f_seed_to_nano_wallet(NANO_PRIVATE_KEY, NANO_PUBLIC_KEY, NANO_SEED, uint32_t);
00645
00655 char *f_nano_key_to_str(char *, unsigned char *);
00656
00675 int f_nano_seed_to_bip39(char *, size_t, size_t *, NANO_SEED, char *);
00676
00691 int f_bip39_to_nano_seed(uint8_t *, char *, char *);
00692
00714 int f_parse_nano_seed_and_bip39_to_JSON(char *, size_t, size_t *, void *, int, const char *);
00715
00733 int f_read_seed(uint8_t *, const char *, void *, int, int);
00734
00749 int f_nano_raw_to_string(char *, size_t *, size_t, void *, int);
00750
00759 int f_nano_valid_nano_str_value(const char *);
00760
00768 int valid_nano_wallet(const char *);
00769
00779 int nano_base_32_2_hex(uint8_t *, char *);
00780
00795 int f_nano_transaction_to_JSON(char *, size_t, size_t *, NANO_PRIVATE_KEY_EXTENDED, F_BLOCK_TRANSFER *);
00796
00804 int valid_raw_balance(const char *);
00805
00813 int is_null_hash(uint8_t *);
00814
00826 int is_nano_prefix(const char *, const char *);
00827
00836 F_FILE_INFO_ERR f_get_nano_file_info(F_NANO_WALLET_INFO *);
00837

```

```

00847 F_FILE_INFO_ERR f_set_nano_file_info(F_NANO_WALLET_INFO *, int);
00848
00872 f_nano_err f_nano_value_compare_value(void *, void *, uint32_t *);
00873
00894 f_nano_err f_nano_verify_nano_funds(void *, void *, void *, uint32_t);
00895
00905 f_nano_err f_nano_parse_raw_str_to_rawl28_t(uint8_t *, const char *);
00906
00916 f_nano_err f_nano_parse_real_str_to_rawl28_t(uint8_t *, const char *);
00917
00937 f_nano_err f_nano_add_sub(void *, void *, void *, uint32_t);
00938
00949 int f_nano_sign_block(F_BLOCK_TRANSFER *, F_BLOCK_TRANSFER *, NANO_PRIVATE_KEY_EXTENDED);
00950
00951 #ifdef __cplusplus
00952 }
00953 #endif
00954

```

## 5.5 f\_util.h File Reference

```

#include <stdint.h>
#include "mbedtls/sha256.h"
#include "mbedtls/aes.h"

```

### Macros

- #define **F\_ENTROPY\_TYPE\_PARANOIC** (uint32\_t)1477682819
- #define **F\_ENTROPY\_TYPE\_EXCELENT** (uint32\_t)1476885281
- #define **F\_ENTROPY\_TYPE\_GOOD** (uint32\_t)1472531015
- #define **F\_ENTROPY\_TYPE\_NOT\_ENOUGH** (uint32\_t)1471001808
- #define **F\_ENTROPY\_TYPE\_NOT\_RECOMENDED** (uint32\_t)1470003345
- #define **ENTROPY\_BEGIN** f\_verify\_system\_entropy\_begin();
- #define **ENTROPY\_END** f\_verify\_system\_entropy\_finish();
- #define **F\_PASS\_MUST\_HAVE\_AT\_LEAST\_NONE** (int)0
- #define **F\_PASS\_MUST\_HAVE\_AT\_LEAST\_ONE\_NUMBER** (int)1
- #define **F\_PASS\_MUST\_HAVE\_AT\_LEAST\_ONE\_SYMBOL** (int)2
- #define **F\_PASS\_MUST\_HAVE\_AT\_LEAST\_ONE\_UPPER\_CASE** (int)4
- #define **F\_PASS\_IS\_TOO\_LONG** (int)256
- #define **F\_PASS\_IS\_TOO\_SHORT** (int)512
- #define **F\_PASS\_IS\_OUT\_OVF** (int)768

### Typedefs

- typedef void(\* **rnd\_fn**) (void \*, size\_t)

### Functions

- int **f\_verify\_system\_entropy** (uint32\_t, void \*, size\_t, int)
- int **f\_pass\_must\_have\_at\_least** (char \*, size\_t, size\_t, size\_t, int)
- void **f\_random\_attach** ( **rnd\_fn**)
- void **f\_random** (void \*, size\_t)
- int **get\_console\_passwd** (char \*, size\_t)

### 5.5.1 Detailed Description

This ABI is a utility for myNanoEmbedded library and sub routines are implemented here.

Definition in file **f\_util.h**.

### 5.5.2 Macro Definition Documentation

#### 5.5.2.1 ENTROPY\_BEGIN

```
#define ENTROPY_BEGIN f_verify_system_entropy_begin();
```

Begins and prepares a entropy function.

See also

**f\_verify\_system\_entropy()** (p. ??)

Definition at line **131** of file **f\_util.h**.

#### 5.5.2.2 ENTROPY\_END

```
#define ENTROPY_END f_verify_system_entropy_finish();
```

Ends a entropy function.

See also

**f\_verify\_system\_entropy()** (p. ??)

Definition at line **138** of file **f\_util.h**.

#### 5.5.2.3 F\_ENTROPY\_TYPE\_EXCELENT

```
#define F_ENTROPY_TYPE_EXCELENT (uint32_t)1476885281
```

Type of the excelent entropy used for verifier.

Slow

Definition at line **103** of file **f\_util.h**.

#### 5.5.2.4 F\_ENTROPY\_TYPE\_GOOD

```
#define F_ENTROPY_TYPE_GOOD (uint32_t)1472531015
```

Type of the good entropy used for verifier.

Not so slow

Definition at line **110** of file **f\_util.h**.

#### 5.5.2.5 F\_ENTROPY\_TYPE\_NOT\_ENOUGH

```
#define F_ENTROPY_TYPE_NOT_ENOUGH (uint32_t)1471001808
```

Type of the moderate entropy used for verifier.

Fast

Definition at line **117** of file **f\_util.h**.

#### 5.5.2.6 F\_ENTROPY\_TYPE\_NOT\_RECOMENDED

```
#define F_ENTROPY_TYPE_NOT_RECOMENDED (uint32_t)1470003345
```

Type of the not recommended entropy used for verifier.

Very fast

Definition at line **124** of file **f\_util.h**.

#### 5.5.2.7 F\_ENTROPY\_TYPE\_PARANOIC

```
#define F_ENTROPY_TYPE_PARANOIC (uint32_t)1477682819
```

Type of the very excelent entropy used for verifier.

Very slow

Definition at line **96** of file **f\_util.h**.

#### 5.5.2.8 F\_PASS\_IS\_OUT\_OVF

```
#define F_PASS_IS_OUT_OVF (int)768
```

Password is overflow and cannot be stored.

Definition at line **180** of file **f\_util.h**.

#### 5.5.2.9 F\_PASS\_IS\_TOO\_LONG

```
#define F_PASS_IS_TOO_LONG (int)256
```

Password is too long.

Definition at line **168** of file **f\_util.h**.

#### 5.5.2.10 F\_PASS\_IS\_TOO\_SHORT

```
#define F_PASS_IS_TOO_SHORT (int)512
```

Password is too short.

Definition at line **174** of file **f\_util.h**.

#### 5.5.2.11 F\_PASS\_MUST\_HAVE\_AT\_LEAST\_NONE

```
#define F_PASS_MUST_HAVE_AT_LEAST_NONE (int)0
```

Password does not need any criteria to pass.

Definition at line **144** of file **f\_util.h**.

#### 5.5.2.12 F\_PASS\_MUST\_HAVE\_AT\_LEAST\_ONE\_NUMBER

```
#define F_PASS_MUST_HAVE_AT_LEAST_ONE_NUMBER (int)1
```

Password must have at least one number.

Definition at line **150** of file **f\_util.h**.



#### 5.5.2.13 F\_PASS\_MUST\_HAVE\_AT\_LEAST\_ONE\_SYMBOL

```
#define F_PASS_MUST_HAVE_AT_LEAST_ONE_SYMBOL (int)2
```

Password must have at least one symbol.

Definition at line **156** of file **f\_util.h**.

#### 5.5.2.14 F\_PASS\_MUST\_HAVE\_AT\_LEAST\_ONE\_UPPER\_CASE

```
#define F_PASS_MUST_HAVE_AT_LEAST_ONE_UPPER_CASE (int)4
```

Password must have at least one upper case.

Definition at line **162** of file **f\_util.h**.

### 5.5.3 Typedef Documentation

#### 5.5.3.1 rnd\_fn

```
rnd_fn
```

Pointer caller for random function.

Definition at line **215** of file **f\_util.h**.

### 5.5.4 Function Documentation

## Parameters

## 5.5.4.1 f\_pass\_must\_have\_at\_least()

```
int f_pass_must_have_at_least (
    char * password,
    size_t n,
    size_t min,
    size_t max,
    int must_have )
```

Checks if a given password has enough requirements to be parsed to a function.

## Parameters

in	<i>password</i>	Password string
in	<i>n</i>	Max buffer string permitted to store password including NULL char
in	<i>min</i>	Minimum size allowed in password string
in	<i>max</i>	Maximum size allowed in password
in	<i>must_have</i>	Must have a type: <ul style="list-style-type: none"> <li>• <code>F_PASS_MUST_HAVE_AT_LEAST_NONE</code> Not need any special characters or number</li> <li>• <code>F_PASS_MUST_HAVE_AT_LEAST_ONE_NUMBER</code> Must have at least one number</li> <li>• <code>F_PASS_MUST_HAVE_AT_LEAST_ONE_SYMBOL</code> Must have at least one symbol</li> <li>• <code>F_PASS_MUST_HAVE_AT_LEAST_ONE_UPPER_CASE</code> Must have at least one upper case</li> </ul>

## Return values:

- `0 (zero)`: If password is passed in the test
- `F_PASS_IS_OUT_OVF`: If password length exceeds *n* value
- `F_PASS_IS_TOO_SHORT`: If password length is less than *min* value
- `F_PASS_IS_TOO_LONG`: If password length is greater than *m* value
- `F_PASS_MUST_HAVE_AT_LEAST_ONE_UPPER_CASE`: If password is required in *must\_have* type upper case characters
- `F_PASS_MUST_HAVE_AT_LEAST_ONE_SYMBOL`: If password is required in *must\_have* type to have symbol(s)
- `F_PASS_MUST_HAVE_AT_LEAST_ONE_NUMBER`: if password is required in *must\_have* type to have number(s)

#### 5.5.4.2 f\_random()

```
void f_random (
    void * random,
    size_t random_sz )
```

Random function to be called to generate a *random* data with *random\_sz*

##### Parameters

out	<i>random</i>	Random data to be parsed
in	<i>random_sz</i>	Size of random data to be filled

##### See also

**f\_random\_attach()** (p. ??)

#### 5.5.4.3 f\_random\_attach()

```
void f_random_attach (
    rnd_fn fn )
```

Attaches a function to be called by **f\_random()** (p. ??)

##### Parameters

in	<i>fn</i>	A function to be called
----	-----------	-------------------------

##### See also

**rnd\_fn** (p. ??)

#### 5.5.4.4 f\_verify\_system\_entropy()

```
int f_verify_system_entropy (
    uint32_t type,
    void * rand,
    size_t rand_sz,
    int turn_on_wdt )
```

Take a random number generator function and returns random value only if randomized data have a desired entropy value.

## Parameters

in	<i>type</i>	Entropy type. Entropy type values are: <ul style="list-style-type: none"> <li>• <code>F_ENTROPY_TYPE_PARANOIC</code> Highest level entropy recommended for generate a Nano SEED with a paranoic entropy. Very slow</li> <li>• <code>F_ENTROPY_TYPE_EXCELENT</code> Gives a very excellent entropy for generating Nano SEED. Slow</li> <li>• <code>F_ENTROPY_TYPE_GOOD</code> Good entropy type for generating Nano SEED. Normal.</li> <li>• <code>F_ENTROPY_TYPE_NOT_ENOUGH</code> Moderate entropy for generating Nano SEED. Usually fast to create a temporary Nano SEED. Fast</li> <li>• <code>F_ENTROPY_TYPE_NOT_RECOMENDED</code> Fast but not recommended for generating Nano SEED.</li> </ul>
out	<i>rand</i>	Random data with a satisfied type of entropy
in	<i>rand_sz</i>	Size of random data output
in	<i>turn_on_wdt</i>	For ESP32, Arduino platform and other microcontrollers only. Turns on/off WATCH DOG (0: OFF, NON ZERO: ON). For Raspberry PI and Linux native is ommited.

## Return values

0	On Success, otherwise Error
---	-----------------------------

5.5.4.5 `get_console_passwd()`

```
int get_console_passwd (
    char * pass,
    size_t pass_sz )
```

Reads a password from console.

## Parameters

out	<i>pass</i>	Password to be parsed to pointer
in	<i>pass_sz</i>	Size of buffer <i>pass</i>

## Return values

0	On Success, otherwise Error
---	-----------------------------

5.6 `f_util.h`

```
00001 /*
```

```

00002     AUTHOR: Fábio Pereira da Silva
00003     YEAR: 2019-20
00004     LICENSE: MIT
00005     EMAIL: fabioegel@gmail.com or fabioegel@protonmail.com
00006 */
00007
00013 #include <stdint.h>
00014 #include "mbedtls/sha256.h"
00015 #include "mbedtls/aes.h"
00016
00017 #ifdef __cplusplus
00018 extern "C" {
00019 #endif
00020
00021 #ifndef F_DOC_SKIP
00022
00023     #define F_LOG_MAX 8*256
00024
00025 #endif
00026
00027 #ifdef F_ESP32
00028
00029     #define F_WDT_MAX_ENTROPY_TIME 2*120
00030     #define F_WDT_PANIC true
00031     #define F_WDT_MIN_TIME 20//4
00032
00033 #endif
00034
00051 int f_verify_system_entropy(uint32_t, void *, size_t, int);
00052
00077 int f_pass_must_have_at_least(char *, size_t, size_t, size_t, int);
00078
00079 #ifndef F_DOC_SKIP
00080
00081 int f_verify_system_entropy_begin();
00082 void f_verify_system_entropy_finish();
00083 int f_file_exists(char *);
00084 int f_find_str(size_t *, char *, size_t, char *);
00085 int f_find_replace(char *, size_t *, size_t, char *, size_t, char *, char *);
00086 int f_is_integer(char *, size_t);
00087 int is_filled_with_value(uint8_t *, size_t, uint8_t);
00088
00089 #endif
00090
00091 // #define F_ENTROPY_TYPE_PARANOIC (uint32_t)1476682819
00096 #define F_ENTROPY_TYPE_PARANOIC (uint32_t)1477682819
00097
00098 // #define F_ENTROPY_TYPE_EXCELENT (uint32_t)1475885281
00103 #define F_ENTROPY_TYPE_EXCELENT (uint32_t)1476885281
00104
00105 // #define F_ENTROPY_TYPE_GOOD (uint32_t)1471531015
00110 #define F_ENTROPY_TYPE_GOOD (uint32_t)1472531015
00111
00112 // #define F_ENTROPY_TYPE_NOT_ENOUGH (uint32_t)1470001808
00117 #define F_ENTROPY_TYPE_NOT_ENOUGH (uint32_t)1471001808
00118
00119 // #define F_ENTROPY_TYPE_NOT_RECOMENDED (uint32_t)1469703345
00124 #define F_ENTROPY_TYPE_NOT_RECOMENDED (uint32_t)1470003345
00125
00131 #define ENTROPY_BEGIN f_verify_system_entropy_begin();
00132
00138 #define ENTROPY_END f_verify_system_entropy_finish();
00139
00144 #define F_PASS_MUST_HAVE_AT_LEAST_NONE (int)0
00145
00150 #define F_PASS_MUST_HAVE_AT_LEAST_ONE_NUMBER (int)1
00151
00156 #define F_PASS_MUST_HAVE_AT_LEAST_ONE_SYMBOL (int)2
00157
00162 #define F_PASS_MUST_HAVE_AT_LEAST_ONE_UPPER_CASE (int)4
00163
00168 #define F_PASS_IS_TOO_LONG (int)256
00169
00174 #define F_PASS_IS_TOO_SHORT (int)512
00175
00180 #define F_PASS_IS_OUT_OVF (int)768
00181
00182 #ifndef F_DOC_SKIP
00183
00184     #define F_PBKDF2_ITER_SZ 2*4096
00185
00186     typedef enum f_pbkdf2_err_t {
00187         F_PBKDF2_RESULT_OK=0,
00188         F_PBKDF2_ERR_CTX=95,
00189         F_PBKDF2_ERR_PKCS5,
00190         F_PBKDF2_ERR_INFO_SHA
00191     } f_pbkdf2_err;

```

```

00192
00193 typedef enum f_aes_err {
00194     F_AES_RESULT_OK=0,
00195     F_AES_ERR_ENCKEY=30,
00196     F_AES_ERR_DECKEY,
00197     F_AES_ERR_MALLOC,
00198     F_AES_UNKNOW_DIRECTION,
00199     F_ERR_ENC_DECRYPT_FAILED
00200 } f_aes_err;
00201
00202 char *fhex2strv2(char *, const void *, size_t, int);
00203 uint8_t *f_sha256_digest(uint8_t *, size_t);
00204 f_pbkdf2_err f_pbkdf2_hmac(unsigned char *, size_t, unsigned char *, size_t, uint8_t *);
00205 f_aes_err f_aes256cipher(uint8_t *, uint8_t *, void *, size_t, void *, int);
00206
00207 #endif
00208
00209 #ifndef F_ESP32
00210
00215 typedef void (*rnd_fn)(void *, size_t);
00216
00224 void f_random_attach(rnd_fn);
00225
00234 void f_random(void *, size_t);
00235
00244 int get_console_passwd(char *, size_t);
00245
00246 #endif
00247
00248 #ifdef __cplusplus
00249 }
00250 #endif

```

## 5.7 sodium.h File Reference

```

#include "sodium/version.h"
#include "sodium/core.h"
#include "sodium/crypto_aead_aes256gcm.h"
#include "sodium/crypto_aead_chacha20poly1305.h"
#include "sodium/crypto_aead_xchacha20poly1305.h"
#include "sodium/crypto_auth.h"
#include "sodium/crypto_auth_hmacsha256.h"
#include "sodium/crypto_auth_hmacsha512.h"
#include "sodium/crypto_auth_hmacsha512256.h"
#include "sodium/crypto_box.h"
#include "sodium/crypto_box_curve25519xsalsa20poly1305.h"
#include "sodium/crypto_core_hsalsa20.h"
#include "sodium/crypto_core_hchacha20.h"
#include "sodium/crypto_core_salsa20.h"
#include "sodium/crypto_core_salsa2012.h"
#include "sodium/crypto_core_salsa208.h"
#include "sodium/crypto_generichash.h"
#include "sodium/crypto_generichash_blake2b.h"
#include "sodium/crypto_hash.h"
#include "sodium/crypto_hash_sha256.h"
#include "sodium/crypto_hash_sha512.h"
#include "sodium/crypto_kdf.h"
#include "sodium/crypto_kdf_blake2b.h"
#include "sodium/crypto_kx.h"
#include "sodium/crypto_onetimeauth.h"
#include "sodium/crypto_onetimeauth_poly1305.h"
#include "sodium/crypto_pwhash.h"
#include "sodium/crypto_pwhash_argon2i.h"
#include "sodium/crypto_scalarmult.h"
#include "sodium/crypto_scalarmult_curve25519.h"
#include "sodium/crypto_secretbox.h"

```

```

#include "sodium/crypto_secretbox_xsalsa20poly1305.h"
#include "sodium/crypto_secretstream_xchacha20poly1305.h"
#include "sodium/crypto_shorthash.h"
#include "sodium/crypto_shorthash_siphhash24.h"
#include "sodium/crypto_sign.h"
#include "sodium/crypto_sign_ed25519.h"
#include "sodium/crypto_stream.h"
#include "sodium/crypto_stream_chacha20.h"
#include "sodium/crypto_stream_salsa20.h"
#include "sodium/crypto_stream_xsalsa20.h"
#include "sodium/crypto_verify_16.h"
#include "sodium/crypto_verify_32.h"
#include "sodium/crypto_verify_64.h"
#include "sodium/randombytes.h"
#include "sodium/randombytes_salsa20_random.h"
#include "sodium/randombytes_sysrandom.h"
#include "sodium/runtime.h"
#include "sodium/utils.h"
#include "sodium/crypto_box_curve25519xchacha20poly1305.h"
#include "sodium/crypto_core_ed25519.h"
#include "sodium/crypto_scalarmult_ed25519.h"
#include "sodium/crypto_secretbox_xchacha20poly1305.h"
#include "sodium/crypto_pwhash_scryptsalsa208sha256.h"
#include "sodium/crypto_stream_salsa2012.h"
#include "sodium/crypto_stream_salsa208.h"
#include "sodium/crypto_stream_xchacha20.h"

```

### 5.7.1 Detailed Description

This header file is an implementation of Libsodium library.

Definition in file **sodium.h**.

## 5.8 sodium.h

```

00001
00005 #ifndef sodium_H
00006 #define sodium_H
00007
00008 #include "sodium/version.h"
00009
00010 #include "sodium/core.h"
00011 #include "sodium/crypto_aead_aes256gcm.h"
00012 #include "sodium/crypto_aead_chacha20poly1305.h"
00013 #include "sodium/crypto_aead_xchacha20poly1305.h"
00014 #include "sodium/crypto_auth.h"
00015 #include "sodium/crypto_auth_hmacsha256.h"
00016 #include "sodium/crypto_auth_hmacsha512.h"
00017 #include "sodium/crypto_auth_hmacsha512256.h"
00018 #include "sodium/crypto_box.h"
00019 #include "sodium/crypto_box_curve25519xsalsa20poly1305.h"
00020 #include "sodium/crypto_core_hsalsa20.h"
00021 #include "sodium/crypto_core_hchacha20.h"
00022 #include "sodium/crypto_core_salsa20.h"
00023 #include "sodium/crypto_core_salsa2012.h"
00024 #include "sodium/crypto_core_salsa208.h"
00025 #include "sodium/crypto_generichash.h"
00026 #include "sodium/crypto_generichash_blake2b.h"
00027 #include "sodium/crypto_hash.h"
00028 #include "sodium/crypto_hash_sha256.h"
00029 #include "sodium/crypto_hash_sha512.h"
00030 #include "sodium/crypto_kdf.h"

```

```
00031 #include "sodium/crypto_kdf_blake2b.h"
00032 #include "sodium/crypto_kx.h"
00033 #include "sodium/crypto_onetimeauth.h"
00034 #include "sodium/crypto_onetimeauth_poly1305.h"
00035 #include "sodium/crypto_pwhash.h"
00036 #include "sodium/crypto_pwhash_argon2i.h"
00037 #include "sodium/crypto_scalarmult.h"
00038 #include "sodium/crypto_scalarmult_curve25519.h"
00039 #include "sodium/crypto_secretbox.h"
00040 #include "sodium/crypto_secretbox_xsalsa20poly1305.h"
00041 #include "sodium/crypto_secretstream_xchacha20poly1305.h"
00042 #include "sodium/crypto_shorthash.h"
00043 #include "sodium/crypto_shorthash_siphhash24.h"
00044 #include "sodium/crypto_sign.h"
00045 #include "sodium/crypto_sign_ed25519.h"
00046 #include "sodium/crypto_stream.h"
00047 #include "sodium/crypto_stream_chacha20.h"
00048 #include "sodium/crypto_stream_salsa20.h"
00049 #include "sodium/crypto_stream_xsalsa20.h"
00050 #include "sodium/crypto_verify_16.h"
00051 #include "sodium/crypto_verify_32.h"
00052 #include "sodium/crypto_verify_64.h"
00053 #include "sodium/randombytes.h"
00054 #ifdef __native_client__
00055 # include "sodium/randombytes_nativeclient.h"
00056 #endif
00057 #include "sodium/randombytes_salsa20_random.h"
00058 #include "sodium/randombytes_sysrandom.h"
00059 #include "sodium/runtime.h"
00060 #include "sodium/utils.h"
00061
00062 #ifndef SODIUM_LIBRARY_MINIMAL
00063 # include "sodium/crypto_box_curve25519xchacha20poly1305.h"
00064 # include "sodium/crypto_core_ed25519.h"
00065 # include "sodium/crypto_scalarmult_ed25519.h"
00066 # include "sodium/crypto_secretbox_xchacha20poly1305.h"
00067 # include "sodium/crypto_pwhash_scryptsalsa208sha256.h"
00068 # include "sodium/crypto_stream_salsa2012.h"
00069 # include "sodium/crypto_stream_salsa208.h"
00070 # include "sodium/crypto_stream_xchacha20.h"
00071 #endif
00072
00073 #endif
```



# Index

\_\_attribute\_\_  
    f\_nano\_crypto\_util.h, 26

account  
    f\_block\_transfer\_t, 7  
    f\_nano\_crypto\_util.h, 43

balance  
    f\_block\_transfer\_t, 7  
    f\_nano\_crypto\_util.h, 43

body  
    f\_nano\_crypto\_util.h, 43  
    f\_nano\_wallet\_info\_t, 15

DEST\_XRB  
    f\_nano\_crypto\_util.h, 20

desc  
    f\_nano\_crypto\_util.h, 43  
    f\_nano\_wallet\_info\_t, 15

description  
    f\_nano\_crypto\_util.h, 44  
    f\_nano\_crypto\_wallet\_t, 10

ENTROPY\_BEGIN  
    f\_util.h, 54

ENTROPY\_END  
    f\_util.h, 54

F\_ADD\_288  
    f\_add\_bn\_288\_le.h, 17

F\_ENTROPY\_TYPE\_EXCELENT  
    f\_util.h, 54

F\_ENTROPY\_TYPE\_GOOD  
    f\_util.h, 54

F\_ENTROPY\_TYPE\_NOT\_ENOUGH  
    f\_util.h, 55

F\_ENTROPY\_TYPE\_NOT\_RECOMENDED  
    f\_util.h, 55

F\_ENTROPY\_TYPE\_PARANOIC  
    f\_util.h, 55

F\_FILE\_INFO\_ERR  
    f\_nano\_crypto\_util.h, 23

F\_PASS\_IS\_OUT\_OVF  
    f\_util.h, 55

F\_PASS\_IS\_TOO\_LONG  
    f\_util.h, 56

F\_PASS\_IS\_TOO\_SHORT  
    f\_util.h, 56

F\_PASS\_MUST\_HAVE\_AT\_LEAST\_NONE  
    f\_util.h, 56

F\_PASS\_MUST\_HAVE\_AT\_LEAST\_ONE\_NUMBER  
    f\_util.h, 56

F\_PASS\_MUST\_HAVE\_AT\_LEAST\_ONE\_SYMBOL  
    f\_util.h, 56

F\_PASS\_MUST\_HAVE\_AT\_LEAST\_ONE\_UPPER\_↵  
    CASE  
    f\_util.h, 57

f\_add\_bn\_288\_le.h, 17, 18

F\_ADD\_288, 17

f\_bip39\_to\_nano\_seed  
    f\_nano\_crypto\_util.h, 26

f\_block\_transfer\_t, 7  
    account, 7  
    balance, 7  
    link, 8  
    preamble, 8  
    prefixes, 8  
    previous, 8  
    representative, 8  
    signature, 9  
    work, 9

f\_cloud\_crypto\_wallet\_nano\_create\_seed  
    f\_nano\_crypto\_util.h, 28

f\_file\_info\_err\_t, 9  
    f\_nano\_crypto\_util.h, 24

f\_generate\_nano\_seed  
    f\_nano\_crypto\_util.h, 29

f\_get\_nano\_file\_info  
    f\_nano\_crypto\_util.h, 29

f\_nano\_add\_sub  
    f\_nano\_crypto\_util.h, 30

f\_nano\_crypto\_util.h, 18, 49  
    \_\_attribute\_\_, 26  
    account, 43  
    balance, 43  
    body, 43  
    DEST\_XRB, 20  
    desc, 43  
    description, 44  
    F\_FILE\_INFO\_ERR, 23  
    f\_bip39\_to\_nano\_seed, 26  
    f\_cloud\_crypto\_wallet\_nano\_create\_seed, 28  
    f\_file\_info\_err\_t, 24  
    f\_generate\_nano\_seed, 29  
    f\_get\_nano\_file\_info, 29  
    f\_nano\_add\_sub, 30  
    f\_nano\_err, 23  
    f\_nano\_err\_t, 25  
    f\_nano\_key\_to\_str, 30  
    f\_nano\_parse\_raw\_str\_to\_raw128\_t, 32

- f\_nano\_parse\_real\_str\_to\_raw128\_t, 32
- f\_nano\_raw\_to\_string, 33
- f\_nano\_seed\_to\_bip39, 33
- f\_nano\_sign\_block, 34
- f\_nano\_transaction\_to\_JSON, 35
- f\_nano\_valid\_nano\_str\_value, 35
- f\_nano\_value\_compare\_value, 36
- f\_nano\_verify\_nano\_funds, 36
- f\_parse\_nano\_seed\_and\_bip39\_to\_JSON, 37
- f\_read\_seed, 38
- f\_seed\_to\_nano\_wallet, 39
- f\_set\_nano\_file\_info, 40
- f\_uint128\_t, 23
- f\_write\_seed\_err, 23
- f\_write\_seed\_err\_t, 26
- file\_info\_integrity, 44
- hash\_sk\_unencrypted, 44
- header, 44
- is\_nano\_prefix, 40
- is\_null\_hash, 41
- iv, 44
- last\_used\_wallet\_number, 45
- link, 45
- MAX\_STR\_NANO\_CHAR, 21
- max\_fee, 45
- NANO\_ENCRYPTED\_SEED\_FILE, 21
- NANO\_FILE\_WALLETS\_INFO, 21
- NANO\_PASSWD\_MAX\_LEN, 21
- NANO\_PREFIX, 21
- NANO\_PRIVATE\_KEY\_EXTENDED, 24
- NANO\_PRIVATE\_KEY, 23
- NANO\_PUBLIC\_KEY\_EXTENDED, 24
- NANO\_PUBLIC\_KEY, 24
- NANO\_SEED, 24
- nano\_base\_32\_2\_hex, 41
- nano\_hdr, 45
- nanoseed\_hash, 45
- PUB\_KEY\_EXTENDED\_MAX\_LEN, 22
- pk\_to\_wallet, 41
- preamble, 46
- prefixes, 46
- previous, 46
- REP\_XRB, 22
- representative, 46
- reserved, 46
- SENDER\_XRB, 22
- STR\_NANO\_SZ, 22
- salt, 47
- seed\_block, 47
- signature, 47
- sk\_encrypted, 47
- sub\_salt, 47
- valid\_nano\_wallet, 42
- valid\_raw\_balance, 42
- ver, 48
- version, 48
- wallet\_prefix, 48
- wallet\_representative, 48
- work, 48
- XRB\_PREFIX, 22
- f\_nano\_crypto\_wallet\_t, 9
  - description, 10
  - iv, 10
  - nano\_hdr, 10
  - salt, 10
  - seed\_block, 11
  - ver, 11
- f\_nano\_encrypted\_wallet\_t, 11
  - hash\_sk\_unencrypted, 12
  - iv, 12
  - reserved, 12
  - sk\_encrypted, 12
  - sub\_salt, 12
- f\_nano\_err
  - f\_nano\_crypto\_util.h, 23
- f\_nano\_err\_t
  - f\_nano\_crypto\_util.h, 25
- f\_nano\_key\_to\_str
  - f\_nano\_crypto\_util.h, 30
- f\_nano\_parse\_raw\_str\_to\_raw128\_t
  - f\_nano\_crypto\_util.h, 32
- f\_nano\_parse\_real\_str\_to\_raw128\_t
  - f\_nano\_crypto\_util.h, 32
- f\_nano\_raw\_to\_string
  - f\_nano\_crypto\_util.h, 33
- f\_nano\_seed\_to\_bip39
  - f\_nano\_crypto\_util.h, 33
- f\_nano\_sign\_block
  - f\_nano\_crypto\_util.h, 34
- f\_nano\_transaction\_to\_JSON
  - f\_nano\_crypto\_util.h, 35
- f\_nano\_valid\_nano\_str\_value
  - f\_nano\_crypto\_util.h, 35
- f\_nano\_value\_compare\_value
  - f\_nano\_crypto\_util.h, 36
- f\_nano\_verify\_nano\_funds
  - f\_nano\_crypto\_util.h, 36
- f\_nano\_wallet\_info\_bdy\_t, 13
  - last\_used\_wallet\_number, 13
  - max\_fee, 13
  - reserved, 14
  - wallet\_prefix, 14
  - wallet\_representative, 14
- f\_nano\_wallet\_info\_t, 14
  - body, 15
  - desc, 15
  - file\_info\_integrity, 15
  - header, 15
  - nanoseed\_hash, 16
  - version, 16
- f\_parse\_nano\_seed\_and\_bip39\_to\_JSON
  - f\_nano\_crypto\_util.h, 37
- f\_pass\_must\_have\_at\_least
  - f\_util.h, 57
- f\_random
  - f\_util.h, 58

- f\_random\_attach
  - f\_util.h, 59
- f\_read\_seed
  - f\_nano\_crypto\_util.h, 38
- f\_seed\_to\_nano\_wallet
  - f\_nano\_crypto\_util.h, 39
- f\_set\_nano\_file\_info
  - f\_nano\_crypto\_util.h, 40
- f\_uint128\_t
  - f\_nano\_crypto\_util.h, 23
- f\_util.h, 53, 60
  - ENTROPY\_BEGIN, 54
  - ENTROPY\_END, 54
  - F\_ENTROPY\_TYPE\_EXCELENT, 54
  - F\_ENTROPY\_TYPE\_GOOD, 54
  - F\_ENTROPY\_TYPE\_NOT\_ENOUGH, 55
  - F\_ENTROPY\_TYPE\_NOT\_RECOMENDED, 55
  - F\_ENTROPY\_TYPE\_PARANOIC, 55
  - F\_PASS\_IS\_OUT\_OVF, 55
  - F\_PASS\_IS\_TOO\_LONG, 56
  - F\_PASS\_IS\_TOO\_SHORT, 56
  - F\_PASS\_MUST\_HAVE\_AT\_LEAST\_NONE, 56
  - F\_PASS\_MUST\_HAVE\_AT\_LEAST\_ONE\_NU↵
    - MBER, 56
  - F\_PASS\_MUST\_HAVE\_AT\_LEAST\_ONE\_SYM↵
    - BOL, 56
  - F\_PASS\_MUST\_HAVE\_AT\_LEAST\_ONE\_UPP↵
    - ER\_CASE, 57
  - f\_pass\_must\_have\_at\_least, 57
  - f\_random, 58
  - f\_random\_attach, 59
  - f\_verify\_system\_entropy, 59
  - get\_console\_passwd, 60
  - rnd\_fn, 57
- f\_verify\_system\_entropy
  - f\_util.h, 59
- f\_write\_seed\_err
  - f\_nano\_crypto\_util.h, 23
- f\_write\_seed\_err\_t
  - f\_nano\_crypto\_util.h, 26
- file\_info\_integrity
  - f\_nano\_crypto\_util.h, 44
  - f\_nano\_wallet\_info\_t, 15
- get\_console\_passwd
  - f\_util.h, 60
- hash\_sk\_unencrypted
  - f\_nano\_crypto\_util.h, 44
  - f\_nano\_encrypted\_wallet\_t, 12
- header
  - f\_nano\_crypto\_util.h, 44
  - f\_nano\_wallet\_info\_t, 15
- is\_nano\_prefix
  - f\_nano\_crypto\_util.h, 40
- is\_null\_hash
  - f\_nano\_crypto\_util.h, 41
- iv
  - f\_nano\_crypto\_util.h, 44
  - f\_nano\_crypto\_wallet\_t, 10
  - f\_nano\_encrypted\_wallet\_t, 12
- last\_used\_wallet\_number
  - f\_nano\_crypto\_util.h, 45
  - f\_nano\_wallet\_info\_bdy\_t, 13
- link
  - f\_block\_transfer\_t, 8
  - f\_nano\_crypto\_util.h, 45
- MAX\_STR\_NANO\_CHAR
  - f\_nano\_crypto\_util.h, 21
- max\_fee
  - f\_nano\_crypto\_util.h, 45
  - f\_nano\_wallet\_info\_bdy\_t, 13
- NANO\_ENCRYPTED\_SEED\_FILE
  - f\_nano\_crypto\_util.h, 21
- NANO\_FILE\_WALLETS\_INFO
  - f\_nano\_crypto\_util.h, 21
- NANO\_PASSWD\_MAX\_LEN
  - f\_nano\_crypto\_util.h, 21
- NANO\_PREFIX
  - f\_nano\_crypto\_util.h, 21
- NANO\_PRIVATE\_KEY\_EXTENDED
  - f\_nano\_crypto\_util.h, 24
- NANO\_PRIVATE\_KEY
  - f\_nano\_crypto\_util.h, 23
- NANO\_PUBLIC\_KEY\_EXTENDED
  - f\_nano\_crypto\_util.h, 24
- NANO\_PUBLIC\_KEY
  - f\_nano\_crypto\_util.h, 24
- NANO\_SEED
  - f\_nano\_crypto\_util.h, 24
- nano\_base\_32\_2\_hex
  - f\_nano\_crypto\_util.h, 41
- nano\_hdr
  - f\_nano\_crypto\_util.h, 45
  - f\_nano\_crypto\_wallet\_t, 10
- nanoseed\_hash
  - f\_nano\_crypto\_util.h, 45
  - f\_nano\_wallet\_info\_t, 16
- PUB\_KEY\_EXTENDED\_MAX\_LEN
  - f\_nano\_crypto\_util.h, 22
- pk\_to\_wallet
  - f\_nano\_crypto\_util.h, 41
- preamble
  - f\_block\_transfer\_t, 8
  - f\_nano\_crypto\_util.h, 46
- prefixes
  - f\_block\_transfer\_t, 8
  - f\_nano\_crypto\_util.h, 46
- previous
  - f\_block\_transfer\_t, 8
  - f\_nano\_crypto\_util.h, 46
- REP\_XRB

- f\_nano\_crypto\_util.h, 22
  - representative
    - f\_block\_transfer\_t, 8
    - f\_nano\_crypto\_util.h, 46
  - reserved
    - f\_nano\_crypto\_util.h, 46
    - f\_nano\_encrypted\_wallet\_t, 12
    - f\_nano\_wallet\_info\_bdy\_t, 14
  - rnd\_fn
    - f\_util.h, 57
- SENDER\_XRB
  - f\_nano\_crypto\_util.h, 22
- STR\_NANO\_SZ
  - f\_nano\_crypto\_util.h, 22
- salt
  - f\_nano\_crypto\_util.h, 47
  - f\_nano\_crypto\_wallet\_t, 10
- seed\_block
  - f\_nano\_crypto\_util.h, 47
  - f\_nano\_crypto\_wallet\_t, 11
- signature
  - f\_block\_transfer\_t, 9
  - f\_nano\_crypto\_util.h, 47
- sk\_encrypted
  - f\_nano\_crypto\_util.h, 47
  - f\_nano\_encrypted\_wallet\_t, 12
- sodium.h, 62, 63
- sub\_salt
  - f\_nano\_crypto\_util.h, 47
  - f\_nano\_encrypted\_wallet\_t, 12
- valid\_nano\_wallet
  - f\_nano\_crypto\_util.h, 42
- valid\_raw\_balance
  - f\_nano\_crypto\_util.h, 42
- ver
  - f\_nano\_crypto\_util.h, 48
  - f\_nano\_crypto\_wallet\_t, 11
- version
  - f\_nano\_crypto\_util.h, 48
  - f\_nano\_wallet\_info\_t, 16
- wallet\_prefix
  - f\_nano\_crypto\_util.h, 48
  - f\_nano\_wallet\_info\_bdy\_t, 14
- wallet\_representative
  - f\_nano\_crypto\_util.h, 48
  - f\_nano\_wallet\_info\_bdy\_t, 14
- work
  - f\_block\_transfer\_t, 9
  - f\_nano\_crypto\_util.h, 48
- XRB\_PREFIX
  - f\_nano\_crypto\_util.h, 22