
py_crypto_hd_wallet

Emanuele Bellocchia

Apr 24, 2024

CONTENTS

1	Introduction	1
2	Install the package	3
3	Test and Coverage	5
4	Modules description	7
5	Examples of wallet JSON outputs	9
6	Documentation	11
7	Buy me a coffee	13
8	License	15
9	Modules	17
9.1	py_crypto_hd_wallet	17
9.1.1	algorand	17
9.1.1.1	hd_wallet_algorand	17
9.1.1.2	hd_wallet_algorand_enum	17
9.1.1.3	hd_wallet_algorand_factory	18
9.1.1.4	hd_wallet_algorand_keys	20
9.1.2	bip	20
9.1.2.1	hd_wallet_bip	20
9.1.2.2	hd_wallet_bip_addr	20
9.1.2.3	hd_wallet_bip_enum	21
9.1.2.4	hd_wallet_bip_factory	22
9.1.2.5	hd_wallet_bip_keys	23
9.1.3	cardano	24
9.1.3.1	shelley	24
9.1.3.1.1	hd_wallet_cardano_shelley	24
9.1.3.1.2	hd_wallet_cardano_shelley_addr	24
9.1.3.1.3	hd_wallet_cardano_shelley_enum	25
9.1.3.1.4	hd_wallet_cardano_shelley_factory	25
9.1.3.1.5	hd_wallet_cardano_shelley_keys	27
9.1.4	common	28
9.1.4.1	hd_wallet_addr_base	28
9.1.4.2	hd_wallet_base	29
9.1.4.3	hd_wallet_data_types	30
9.1.4.4	hd_wallet_enum_dict	31

9.1.4.5	hd_wallet_keys_base	31
9.1.5	electrum	32
9.1.5.1	v1	32
9.1.5.1.1	hd_wallet_electrum_v1	32
9.1.5.1.2	hd_wallet_electrum_v1_addr	33
9.1.5.1.3	hd_wallet_electrum_v1_enum	33
9.1.5.1.4	hd_wallet_electrum_v1_factory	34
9.1.5.1.5	hd_wallet_electrum_v1_keys	35
9.1.5.2	v2	36
9.1.5.2.1	hd_wallet_electrum_v2	36
9.1.5.2.2	hd_wallet_electrum_v2_addr	37
9.1.5.2.3	hd_wallet_electrum_v2_enum	37
9.1.5.2.4	hd_wallet_electrum_v2_factory	38
9.1.5.2.5	hd_wallet_electrum_v2_keys	39
9.1.6	monero	40
9.1.6.1	hd_wallet_monero	40
9.1.6.2	hd_wallet_monero_enum	40
9.1.6.3	hd_wallet_monero_factory	41
9.1.6.4	hd_wallet_monero_keys	43
9.1.6.5	hd_wallet_monero_subaddr	43
9.1.7	saver	43
9.1.7.1	hd_wallet_saver	43
9.1.8	substrate	44
9.1.8.1	hd_wallet_substrate	44
9.1.8.2	hd_wallet_substrate_enum	44
9.1.8.3	hd_wallet_substrate_factory	45
9.1.8.4	hd_wallet_substrate_keys	46
9.1.9	utils	47
9.1.9.1	utils	47

Python Module Index **49**

Index **51**

INTRODUCTION

This package contains a very basic implementation of a HD (Hierarchical Deterministic) wallet based on my [bip_utils](#) library. It is basically a nice wrapper for the *bip_utils* library for generating mnemonics, seeds, public/private keys and addresses. Therefore, it has no network functionalities. The supported coins are the same of the [bip_utils](#) library, so check the related page.

INSTALL THE PACKAGE

The package requires Python 3, it is not compatible with Python 2. To install it:

- Using *pip*, from this directory (local):

```
pip install .
```

- Using *pip*, from PyPI:

```
pip install py_crypto_hd_wallet
```

NOTE: if you are using an Apple M1, please make sure to update *coincurve* (required by *bip_utils*) to version 17.0.0 otherwise it won't work.

TEST AND COVERAGE

Install develop dependencies:

```
pip install -r requirements-dev.txt
```

To run tests:

```
python -m unittest discover
```

To run tests with coverage:

```
coverage run -m unittest discover  
coverage report
```

To run code analysis, just execute the `analyze_code` script.

MODULES DESCRIPTION

- BIP wallet
- Algorand wallet
- Cardano Shelley wallet
- Electrum V1 wallet
- Electrum V2 wallet
- Monero wallet
- Substrate wallet

EXAMPLES OF WALLET JSON OUTPUTS

- [BIP wallet](#)
- [Algorand wallet](#)
- [Cardano Shelley wallet](#)
- [Electrum V1 wallet](#)
- [Electrum V2 wallet](#)
- [Monero wallet](#)
- [Substrate wallet](#)

DOCUMENTATION

The library documentation is available at py-crypto-hd-wallet.readthedocs.io.

BUY ME A COFFEE

You know, I'm italian and I love drinking coffee (especially while coding :D). So, if you'd like to buy me one:

- BTC: `bc1qq4r9cglwzd6f2hxxvdkucmdejvr9h8me5hy0k8`
- ERC20/BEP20: `0xf84e4898E5E10bf1fBe9ffA3EEC845e82e364b5B`

Thank you very much for your support.

LICENSE

This software is available under the MIT license.

MODULES

9.1 py_crypto_hd_wallet

9.1.1 algorand

9.1.1.1 hd_wallet_algorand

Module for generating Algorand wallets.

class HdWalletAlgorand(*wallet_name: str, bip_obj: Bip44Base, mnemonic: str = "", seed_bytes: bytes = b"*)

Bases: *HdWalletBase*

HD wallet Algorand class. It allows to generate an Algorand wallet like the official one.

m_bip_obj: *Bip44Base*

Generate(***kwargs: Any*) → None

Generate wallet keys and addresses.

IsWatchOnly() → bool

Get if the wallet is watch-only.

Returns :

bool: True if watch-only, false otherwise

9.1.1.2 hd_wallet_algorand_enum

Module with enums for Algorand wallets.

class HdWalletAlgorandDataTypes(*value*)

Bases: *HdWalletDataTypes*

Enumerative for wallet Algorand data types.

WALLET_NAME = 1

COIN_NAME = 2

MNEMONIC = 3

SEED_BYTES = 4

KEY = 5

```
class HdWalletAlgorandKeyTypes(value)
```

Bases: *HdWalletKeyTypes*

Enumerative for wallet Algorand key types.

PRIV = 1

PUB = 2

ADDRESS = 3

9.1.1.3 hd_wallet_algorand_factory

Module for creating Algorand wallet factories.

```
class HdWalletAlgorandFactory
```

Bases: object

HD wallet Algorand factory class. It allows a HdWalletAlgorand to be created in different way.

```
CreateRandom(wallet_name: str, words_num: AlgorandWordsNum =  
                AlgorandWordsNum.WORDS_NUM_25, lang: AlgorandLanguages =  
                AlgorandLanguages.ENGLISH) → HdWalletBase
```

Create wallet randomly.

Parameters

- **wallet_name** (*str*) – Wallet name
- **words_num** (*HdWalletAlgorandWordsNum*, *optional*) – Words number (default: 25)
- **lang** (*HdWalletAlgorandLanguages*, *optional*) – Language (default: English)

Returns

HdWalletBase object

Return type

HdWalletBase object

Raises

TypeError – If words number is not a *HdWalletAlgorandWordsNum* enum or language is not a *HdWalletAlgorandLanguages* enum

```
CreateFromMnemonic(wallet_name: str, mnemonic: str) → HdWalletBase
```

Create wallet from mnemonic.

Parameters

- **wallet_name** (*str*) – Wallet name
- **mnemonic** (*str*) – Mnemonic

Returns

HdWalletBase object

Return type

HdWalletBase object

Raises

ValueError – If the mnemonic is not valid

CreateFromSeed(*wallet_name: str, seed_bytes: bytes*) → *HdWalletBase*

Create wallet from seed.

Parameters

- **wallet_name** (*str*) – Wallet name
- **seed_bytes** (*bytes*) – Seed bytes

Returns

HdWalletBase object

Return type

HdWalletBase object

Raises

ValueError – If the seed is not valid

CreateFromPrivateKey(*wallet_name: str, priv_key_bytes: bytes*) → *HdWalletBase*

Create wallet from private key.

Parameters

- **wallet_name** (*str*) – Wallet name
- **priv_key_bytes** (*bytes*) – Private key bytes

Returns

HdWalletBase object

Return type

HdWalletBase object

Raises

ValueError – If the private key is not valid

CreateFromPublicKey(*wallet_name: str, pub_key_bytes: bytes*) → *HdWalletBase*

Create wallet from public key.

Parameters

- **wallet_name** (*str*) – Wallet name
- **pub_key_bytes** (*bytes*) – Public key bytes

Returns

HdWalletBase object

Return type

HdWalletBase object

Raises

ValueError – If the public key is not valid

9.1.1.4 hd_wallet_algorand_keys

Module with helper class for storing Algorand keys.

class HdWalletAlgorandKeys(*bip_obj: Bip44Base*)

Bases: [HdWalletKeysBase](#)

HD wallet Algorand keys class. It creates keys from an Algorand object and stores them. Keys can be got individually, as dictionary or in JSON format.

m_key_enum: `Type[Enum]`

m_dict_data: `Dict[str, Any]`

9.1.2 bip

9.1.2.1 hd_wallet_bip

Module for generating wallets based on BIP specifications.

class HdWalletBip(*wallet_name: str, bip_obj: Bip44Base, mnemonic: str = "", passphrase: str = "", seed_bytes: bytes = b""*)

Bases: [HdWalletBase](#)

HD wallet BIP class. It allows to generate a complete wallet based on BIP specifications.

m_bip_obj: `Bip44Base`

Generate(***kwargs: Any*) → None

Generate wallet keys and addresses.

Parameters

- **acc_idx** (*int, optional*) – Account index (default: 0)
- **change_idx** (*HdWalletBipChanges, optional*) – Change index (default: external)
- **addr_num** (*int, optional*) – Number of addresses to be generated (default: 20)
- **addr_off** (*int, optional*) – Starting address index (default: 0)

IsWatchOnly() → bool

Get if the wallet is watch-only.

Returns :

bool: True if watch-only, false otherwise

9.1.2.2 hd_wallet_bip_addr

Module with helper class for storing BIP addresses.

class HdWalletBipAddresses(*bip_obj: Bip44Base, addr_num: int, addr_off: int*)

Bases: [HdWalletAddrBase](#)

HD wallet BIP addresses class. It creates addresses from a Bip object and stores them. Addresses can be got individually, as dictionary or in JSON format.

m_addr_off: `int`


```

m_addr: List[Any]

m_dict_key_str_format: str

```

9.1.2.3 hd_wallet_bip_enum

Module with enums for BIP wallets.

```

class HdWalletBipDataTypes(value)
    Bases: HdWalletDataTypes
    Enumerative for wallet BIP data types.

    WALLET_NAME = 1

    COIN_NAME = 2

    SPEC_NAME = 3

    MNEMONIC = 4

    PASSPHRASE = 5

    SEED_BYTES = 6

    ACCOUNT_IDX = 7

    CHANGE_IDX = 8

    MASTER_KEY = 9

    PURPOSE_KEY = 10

    COIN_KEY = 11

    ACCOUNT_KEY = 12

    CHANGE_KEY = 13

    ADDRESS_OFF = 14

    ADDRESS = 15

class HdWalletBipKeyTypes(value)
    Bases: HdWalletKeyTypes
    Enumerative for wallet BIP key types.

    EX_PRIV = 1

    RAW_PRIV = 2

    WIF_PRIV = 3

    EX_PUB = 4

    RAW_COMPR_PUB = 5

    RAW_UNCOMPR_PUB = 6

    ADDRESS = 7

```

9.1.2.4 hd_wallet_bip_factory

Module for creating BIP wallet factories.

class HdWalletBipFactoryConst

Bases: object

Class container for HD wallet BIP factory constants.

```
BIP_COIN_TO_CLASS: Dict[Type[BipCoins], Type[Bip44Base]] = {<enum 'Bip44Coins'>:
<class 'bip_utils.bip.bip44.bip44.Bip44'>, <enum 'Bip49Coins'>: <class
'bip_utils.bip.bip49.bip49.Bip49'>, <enum 'Bip84Coins'>: <class
'bip_utils.bip.bip84.bip84.Bip84'>, <enum 'Bip86Coins'>: <class
'bip_utils.bip.bip86.bip86.Bip86'>}
```

class HdWalletBipFactory(*coin_type: BipCoins*)

Bases: object

HD wallet BIP factory class. It allows a HdWalletBip to be created in different ways.

m_bip_cls: Type[Bip44Base]

m_bip_coin: BipCoins

CreateRandom(*wallet_name: str, words_num: Bip39WordsNum = Bip39WordsNum.WORDS_NUM_24,*
lang: Bip39Languages = Bip39Languages.ENGLISH) → *HdWalletBase*

Create wallet randomly.

Parameters

- **wallet_name** (*str*) – Wallet name
- **words_num** (*HdWalletBipWordsNum, optional*) – Words number (default: 24)
- **lang** (*HdWalletBipLanguages, optional*) – Language (default: English)

Returns

HdWalletBase object

Return type

HdWalletBase object

Raises

TypeError – If words number is not a HdWalletBipWordsNum enum or language is not a HdWalletBipLanguages enum

CreateFromMnemonic(*wallet_name: str, mnemonic: str, passphrase: str = ""*) → *HdWalletBase*

Create wallet from mnemonic.

Parameters

- **wallet_name** (*str*) – Wallet name
- **mnemonic** (*str*) – Mnemonic
- **passphrase** (*str, optional*) – Passphrase for protecting mnemonic, empty if not specified

Returns

HdWalletBase object

Return type

HdWalletBase object

Raises

ValueError – If the mnemonic is not valid

CreateFromSeed(*wallet_name: str, seed_bytes: bytes*) → *HdWalletBase*

Create wallet from seed.

Parameters

- **wallet_name** (*str*) – Wallet name
- **seed_bytes** (*bytes*) – Seed bytes

Returns

HdWalletBase object

Return type

HdWalletBase object

Raises

ValueError – If the seed is not valid

CreateFromExtendedKey(*wallet_name: str, ex_key_str: str*) → *HdWalletBase*

Create wallet from extended key.

Parameters

- **wallet_name** (*str*) – Wallet name
- **ex_key_str** (*str*) – Extended key string

Returns

HdWalletBase object

Return type

HdWalletBase object

Raises

ValueError – If the extended key is not valid

9.1.2.5 hd_wallet_bip_keys

Module with helper class for storing BIP keys.

class HdWalletBipKeys(*bip_obj: Bip44Base*)

Bases: *HdWalletKeysBase*

HD wallet BIP keys class. It creates keys from a Bip object and stores them. Keys can be got individually, as dictionary or in JSON format.

m_key_enum: `Type[Enum]`

m_dict_data: `Dict[str, Any]`

9.1.3 cardano

9.1.3.1 shelley

9.1.3.1.1 hd_wallet_cardano_shelley

Module for generating wallets based on Cardano Shelley.

class HdWalletCardanoShelley(*wallet_name: str, bip_obj: Bip44Base, mnemonic: str = "", passphrase: str = "", seed_bytes: bytes = b"*)

Bases: [*HdWalletBase*](#)

HD wallet Cardano Shelley class. It allows to generate a complete wallet based on Cardano Shelley specifications.

m_bip_obj: [**Bip44Base**](#)

Generate(***kwargs: Any*) → None

Generate wallet keys and addresses.

Parameters

- **acc_idx** (*int, optional*) – Account index (default: 0)
- **change_idx** (*HdWalletCardanoShelleyChanges, optional*) – Change index (default: external)
- **addr_num** (*int, optional*) – Number of addresses to be generated (default: 20)
- **addr_off** (*int, optional*) – Starting address index (default: 0)

IsWatchOnly() → bool

Get if the wallet is watch-only.

Returns :

bool: True if watch-only, false otherwise

9.1.3.1.2 hd_wallet_cardano_shelley_addr

Module with helper class for storing Cardano Shelley addresses.

class HdWalletCardanoShelleyAddresses(*shelley_obj: CardanoShelley, addr_num: int, addr_off: int*)

Bases: [*HdWalletAddrBase*](#)

HD wallet Cardano Shelley addresses class. It creates addresses from a CardanoShelley object and stores them. Addresses can be got individually, as dictionary or in JSON format.

m_addr_off: **int**

m_addr: **List[*Any*]**

m_dict_key_str_format: **str**

9.1.3.1.3 hd_wallet_cardano_shelley_enum

Module with enums for Cardano Shelley wallets.

class HdWalletCardanoShelleyDataTypes(*value*)

Bases: *HdWalletDataTypes*

Enumerative for wallet Cardano Shelley data types.

WALLET_NAME = 1

COIN_NAME = 2

MNEMONIC = 3

PASSPHRASE = 4

SEED_BYTES = 5

ACCOUNT_IDX = 6

CHANGE_IDX = 7

MASTER_KEY = 8

ACCOUNT_KEY = 9

STAKING_KEY = 10

ADDRESS_OFF = 11

ADDRESS = 12

class HdWalletCardanoShelleyKeyTypes(*value*)

Bases: *HdWalletKeyTypes*

Enumerative for wallet Cardano Shelley key types.

RAW_PRIV = 1

RAW_PUB = 2

ADDRESS = 3

9.1.3.1.4 hd_wallet_cardano_shelley_factory

Module for creating Cardano Shelley wallet factories.

class HdWalletCardanoShelleyFactory(*coin_type: Cip1852Coins*)

Bases: object

HD wallet Cardano Shelley factory class. It allows a HdWalletCardanoShelley to be created in different ways.

m_coin: **Cip1852Coins**

CreateRandom(*wallet_name: str, words_num: Optional[Bip39WordsNum] = None, lang: Bip39Languages = Bip39Languages.ENGLISH*) → *HdWalletBase*

Create wallet randomly.

Parameters

- **wallet_name** (*str*) – Wallet name
- **words_num** (*HdWalletCardanoShelleyWordsNum, optional*) – Words number (default: 15 for Icaurs, 24 for Ledger)
- **lang** (*HdWalletCardanoShelleyLanguages, optional*) – Language (default: English)

Returns

HdWalletBase object

Return type

HdWalletBase object

Raises

TypeError – If words number is not a *HdWalletBipWordsNum* enum or language is not a *HdWalletBipLanguages* enum

CreateFromMnemonic(*wallet_name: str, mnemonic: str, passphrase: str = ""*) → *HdWalletBase*

Create wallet from mnemonic.

Parameters

- **wallet_name** (*str*) – Wallet name
- **mnemonic** (*str*) – Mnemonic
- **passphrase** (*str, optional*) – Passphrase for protecting mnemonic, empty if not specified

Returns

HdWalletBase object

Return type

HdWalletBase object

Raises

ValueError – If the mnemonic is not valid

CreateFromSeed(*wallet_name: str, seed_bytes: bytes*) → *HdWalletBase*

Create wallet from seed.

Parameters

- **wallet_name** (*str*) – Wallet name
- **seed_bytes** (*bytes*) – Seed bytes

Returns

HdWalletBase object

Return type

HdWalletBase object

Raises

ValueError – If the seed is not valid

CreateFromPrivateKey(*wallet_name: str, priv_key_bytes: bytes*) → *HdWalletBase*

Create wallet from private key. The key will be considered a master key.

Parameters

- **wallet_name** (*str*) – Wallet name
- **priv_key_bytes** (*bytes*) – Private key bytes

Returns

HdWalletBase object

Return type

HdWalletBase object

Raises

ValueError – If the private key is not valid

CreateFromPublicKey(*wallet_name: str, pub_key_bytes: bytes*) → *HdWalletBase*

Create wallet from public key. The key will be considered an account key in order to derive child keys.

Parameters

- **wallet_name** (*str*) – Wallet name
- **pub_key_bytes** (*bytes*) – Public key bytes

Returns

HdWalletBase object

Return type

HdWalletBase object

Raises

ValueError – If the public key is not valid

9.1.3.1.5 hd_wallet_cardano_shelley_keys

Module with helper class for storing Cardano Shelley keys.

class HdWalletCardanoShelleyKeysBase(*key_enum: Type[Enum]*)

Bases: *HdWalletKeysBase*

HD wallet Cardano Shelley keys base class. It contains some helper methods for Cardano Shelley keys classes.

m_key_enum: *Type[Enum]*

m_dict_data: *Dict[str, Any]*

class HdWalletCardanoShelleyMasterKeys(*bip_obj: Bip44Base*)

Bases: *HdWalletCardanoShelleyKeysBase*

HD wallet Cardano Shelley master keys class. It creates keys from a CardanoShelley object and stores them. Keys can be got individually, as dictionary or in JSON format.

m_key_enum: *Type[Enum]*

m_dict_data: *Dict[str, Any]*

class HdWalletCardanoShelleyStakingKeys(*shelley_obj: CardanoShelley*)

Bases: [HdWalletCardanoShelleyKeysBase](#)

HD wallet Cardano Shelley staking keys class. It creates keys from a CardanoShelley object and stores them. Keys can be got individually, as dictionary or in JSON format.

m_key_enum: Type[Enum]

m_dict_data: Dict[str, Any]

class HdWalletCardanoShelleyDerivedKeys(*shelley_obj: CardanoShelley*)

Bases: [HdWalletCardanoShelleyKeysBase](#)

HD wallet Cardano Shelley derived keys class. It creates keys from a CardanoShelley object and stores them. Keys can be got individually, as dictionary or in JSON format.

m_key_enum: Type[Enum]

m_dict_data: Dict[str, Any]

9.1.4 common

9.1.4.1 hd_wallet_addr_base

Module with base class for wallet addresses.

class HdWalletAddrBaseConst

Bases: object

Class container for HD wallet addresses base constants.

DICT_KEY_DEF_FORMAT: str = 'address_{:d}'

class HdWalletAddrBase(*addr_off: int, dict_key_str_format: Optional[str] = None*)

Bases: ABC

HD wallet addresses base class. It shall be inherited by wallet addresses classes.

m_addr_off: int

m_addr: List[Any]

m_dict_key_str_format: str

ToDict() → Dict[str, Any]

Get addresses as a dictionary.

Returns

Addresses as a dictionary

Return type

dict

ToJson(*json_indent: int = 4*) → str

Get addresses as string in JSON format.

Parameters

json_indent (*int, optional*) – Indent for JSON format, 4 by default

Returns

Addresses as string in JSON format

Return type

str

Count() → int

Get the addresses count.

Returns

Number of addresses

Return type

int

__getitem__(*addr_idx: int*) → Any

Get the specified address index.

Parameters**addr_idx** (*int*) – Address index**Returns**

Address

Return type

Any

__iter__() → Iterator[str]

Get the iterator to the current element.

Returns

Iterator to the current element

Return type

Iterator object

9.1.4.2 hd_wallet_base

Module with base class for wallet generators.

class HdWalletBase(*key_enum: Type[Enum]*)Bases: *HdWalletEnumDict*, ABC

HD wallet base class. It shall be inherited by wallet classes.

abstract Generate(***kwargs: Any*) → None

Generate wallet keys and addresses.

Parameters****kwargs** – Arbitrary arguments depending on the wallet type**abstract IsWatchOnly**() → bool

Get if the wallet is watch-only.

Returns :

bool: True if watch-only, false otherwise

ToDict() → Dict[str, Any]

Get wallet data as a dictionary.

Returns

Wallet data as a dictionary

Return type

dict

HasData(*key*: HdWalletDataTypes) → bool

Get if the specified key is present.

Parameters

key (HdWalletKeyTypes) – Key

Returns

True if present, false otherwise

Return type

bool

Raises

TypeError – If the enumerative is not of the correct type

GetData(*key*: HdWalletDataTypes) → Optional[Any]

Get the specified key value.

Parameters

key (HdWalletKeyTypes) – Key

Returns

Key value None: If the key type is not found

Return type

str

Raises

TypeError – If the enumerative is not of the correct type

m_key_enum: Type[Enum]

m_dict_data: Dict[str, Any]

9.1.4.3 hd_wallet_data_types

Module for generic wallet data types enum.

class HdWalletDataTypes(*value*)

Bases: Enum

Base enum for wallet data types.

class HdWalletKeyTypes(*value*)

Bases: Enum

Base enum for wallet key types.

9.1.4.4 hd_wallet_enum_dict

Module with enum dictionary class.

class HdWalletEnumDict(*key_enum: Type[Enum]*)

Bases: object

HD wallet enum dictionary class. It allows a dictionary to be accessed using enum as string keys. It shall be inherited by all classes needed to use dictionaries to be accessed by enums.

m_key_enum: Type[Enum]

m_dict_data: Dict[str, Any]

KeyEnum() → Type[Enum]

Get key enumerative type.

Returns

Key enumerative type

Return type

Enum

ToDict() → Dict[str, Any]

Get keys as a dictionary.

Returns

Keys as a dictionary

Return type

dict

ToJson(*json_indent: int = 4*) → str

Get keys as string in JSON format.

Parameters

json_indent (*int*, *optional*) – Indent for JSON format, 4 by default

Returns

Keys as string in JSON format

Return type

str

9.1.4.5 hd_wallet_keys_base

Module with base class for wallet keys.

class HdWalletKeysBase(*key_enum: Type[Enum]*)

Bases: [HdWalletEnumDict](#)

HD wallet keys base class. It shall be inherited by wallet keys classes.

HasKey(*key: HdWalletKeyTypes*) → bool

Get if the specified key is present.

Parameters

key ([HdWalletKeyTypes](#)) – Key

Returns

True if present, false otherwise

Return type

bool

Raises**TypeError** – If the enumerative is not of the correct type**GetKey**(*key*: HdWalletKeyTypes) → Optional[str]

Get the specified key value.

Parameters**key** (HdWalletKeyTypes) – Key**Returns**

Key value None: If the key type is not found

Return type

str

Raises**TypeError** – If the enumerative is not of the correct type**m_key_enum**: Type[Enum]**m_dict_data**: Dict[str, Any]

9.1.5 electrum

9.1.5.1 v1

9.1.5.1.1 hd_wallet_electrum_v1

Module for generating wallets based on Electrum V1.

class HdWalletElectrumV1(*wallet_name*: str, *electrum_obj*: ElectrumV1, *mnemonic*: str = "", *seed_bytes*: bytes = b'')

Bases: HdWalletBase

HD wallet Electrum V1 class. It allows to generate a wallet like Electrum V1.

m_electrum_obj: ElectrumV1**Generate**(***kwargs*: Any) → None

Generate wallet keys and addresses.

Parameters

- **change_idx** (*int*, *optional*) – Change index (default: 0)
- **addr_num** (*int*, *optional*) – Number of addresses to be generated (default: 20)
- **addr_off** (*int*, *optional*) – Starting address index (default: 0)

IsWatchOnly() → bool

Get if the wallet is watch-only.

Returns :

bool: True if watch-only, false otherwise

9.1.5.1.2 hd_wallet_electrum_v1_addr

Module with helper class for storing Electrum V1 addresses.

```
class HdWalletElectrumV1Addresses(electrum_obj: ElectrumV1, change_idx: int, addr_num: int, addr_off:  
int)
```

Bases: *HdWalletAddrBase*

HD wallet Electrum V1 addresses class. It creates addresses from an ElectrumV1 object and stores them. Addresses can be got individually, as dictionary or in JSON format.

m_addr_off: int

m_addr: List[Any]

m_dict_key_str_format: str

9.1.5.1.3 hd_wallet_electrum_v1_enum

Module with enums for Electrum V1 wallets.

```
class HdWalletElectrumV1DataTypes(value)
```

Bases: *HdWalletDataTypes*

Enumerative for wallet Electrum V1 data types.

WALLET_NAME = 1

COIN_NAME = 2

MNEMONIC = 3

SEED_BYTES = 4

CHANGE_IDX = 5

MASTER_KEY = 6

ADDRESS_OFF = 7

ADDRESS = 8

```
class HdWalletElectrumV1KeyTypes(value)
```

Bases: *HdWalletKeyTypes*

Enumerative for wallet Electrum V1 key types.

RAW_PRIV = 1

WIF_PRIV = 2

RAW_PUB = 3

ADDRESS = 4

9.1.5.1.4 hd_wallet_electrum_v1_factory

Module for creating Electrum V1 wallet factories.

class HdWalletElectrumV1Factory

Bases: object

HD wallet Electrum V1 factory class. It allows a HdWalletElectrumV1 to be created in different ways.

CreateRandom(*wallet_name: str, words_num: ElectrumV1WordsNum = ElectrumV1WordsNum.WORDS_NUM_12, lang: ElectrumV1Languages = ElectrumV1Languages.ENGLISH*) → *HdWalletBase*

Create wallet randomly.

Parameters

- **wallet_name** (*str*) – Wallet name
- **words_num** (*HdWalletElectrumV1WordsNum, optional*) – Words number (default: 12)
- **lang** (*HdWalletElectrumV1Languages, optional*) – Language (default: English)

Returns

HdWalletBase object

Return type

HdWalletBase object

Raises

TypeError – If words number is not a HdWalletElectrumV1WordsNum enum or language is not a HdWalletElectrumV1Languages enum

CreateFromMnemonic(*wallet_name: str, mnemonic: str*) → *HdWalletBase*

Create wallet from mnemonic.

Parameters

- **wallet_name** (*str*) – Wallet name
- **mnemonic** (*str*) – Mnemonic

Returns

HdWalletBase object

Return type

HdWalletBase object

Raises

ValueError – If the mnemonic is not valid

CreateFromSeed(*wallet_name: str, seed_bytes: bytes*) → *HdWalletBase*

Create wallet from seed.

Parameters

- **wallet_name** (*str*) – Wallet name
- **seed_bytes** (*bytes*) – Seed bytes

Returns

HdWalletBase object

Return type

HdWalletBase object

Raises**ValueError** – If the seed is not valid**CreateFromPrivateKey**(*wallet_name: str, priv_key_bytes: bytes*) → *HdWalletBase*

Create wallet from private key.

Parameters

- **wallet_name** (*str*) – Wallet name
- **priv_key_bytes** (*bytes*) – Private key bytes

Returns

HdWalletBase object

Return type

HdWalletBase object

Raises**ValueError** – If the private key is not valid**CreateFromPublicKey**(*wallet_name: str, pub_key_bytes: bytes*) → *HdWalletBase*

Create wallet from public key.

Parameters

- **wallet_name** (*str*) – Wallet name
- **pub_key_bytes** (*bytes*) – Public key bytes

Returns

HdWalletBase object

Return type

HdWalletBase object

Raises**ValueError** – If the private key is not valid

9.1.5.1.5 hd_wallet_electrum_v1_keys

Module with helper class for storing Electrum V1 keys.

class HdWalletElectrumV1KeyUtils

Bases: object

Class container for HD wallet Electrum keys utility functions.

static PrivToWif(*priv_key: IPrivateKey*) → str

Encode private key to WIF.

Parameters**priv_key** (*IPrivateKey object*) – Private key**Returns**

WIF encoded key

Return type

str

class HdWalletElectrumV1MasterKeys(*electrum_obj: ElectrumV1*)

Bases: *HdWalletKeysBase*

HD wallet Electrum master keys class. It creates master keys from an ElectrumV1 object and stores them. Keys can be got individually, as dictionary or in JSON format.

m_key_enum: Type[Enum]

m_dict_data: Dict[str, Any]

class HdWalletElectrumV1DerivedKeys(*electrum_obj: ElectrumV1, change_idx: int, addr_num: int, addr_off: int*)

Bases: *HdWalletKeysBase*

HD wallet Electrum derived keys class. It creates derived from an Electrum object and stores them. Keys can be got individually, as dictionary or in JSON format.

m_key_enum: Type[Enum]

m_dict_data: Dict[str, Any]

9.1.5.2 v2

9.1.5.2.1 hd_wallet_electrum_v2

Module for generating wallets based on Electrum V2.

class HdWalletElectrumV2(*wallet_name: str, electrum_obj: ElectrumV2Base, mnemonic: str = "", passphrase: str = "", seed_bytes: bytes = b"*)

Bases: *HdWalletBase*

HD wallet Electrum V2 class. It allows to generate a wallet like Electrum V2.

m_electrum_obj: ElectrumV2Base

Generate(***kwargs: Any*) → None

Generate wallet keys and addresses.

Parameters

- **change_idx** (*int, optional*) – Change index (default: 0)
- **addr_num** (*int, optional*) – Number of addresses to be generated (default: 20)
- **addr_off** (*int, optional*) – Starting address index (default: 0)

IsWatchOnly() → bool

Get if the wallet is watch-only.

Returns :

bool: True if watch-only, false otherwise

9.1.5.2.2 hd_wallet_electrum_v2_addr

Module with helper class for storing Electrum V2 addresses.

```
class HdWalletElectrumV2Addresses(electrum_obj: ElectrumV2Base, change_idx: int, addr_num: int,  
                                addr_off: int)
```

Bases: *HdWalletAddrBase*

HD wallet Electrum V2 addresses class. It creates addresses from an ElectrumV2 object and stores them. Addresses can be got individually, as dictionary or in JSON format.

```
m_addr_off:  int  
m_addr:    List[Any]  
m_dict_key_str_format:  str
```

9.1.5.2.3 hd_wallet_electrum_v2_enum

Module with enums for Electrum V2 wallets.

```
class HdWalletElectrumV2DataTypes(value)
```

Bases: *HdWalletDataTypes*

Enumerative for wallet Electrum V2 data types.

```
WALLET_NAME = 1  
COIN_NAME = 2  
MNEMONIC = 3  
PASSPHRASE = 4  
SEED_BYTES = 5  
CHANGE_IDX = 6  
MASTER_KEY = 7  
ADDRESS_OFF = 8  
ADDRESS = 9
```

```
class HdWalletElectrumV2KeyTypes(value)
```

Bases: *HdWalletKeyTypes*

Enumerative for wallet Electrum V2 key types.

```
EX_PRIV = 1  
RAW_PRIV = 2  
WIF_PRIV = 3  
EX_PUB = 4  
RAW_PUB = 5  
ADDRESS = 6
```

9.1.5.2.4 hd_wallet_electrum_v2_factory

Module for creating Electrum V2 wallet factories.

class HdWalletElectrumV2Factory(*mnemonic_type: ElectrumV2MnemonicTypes*)

Bases: object

HD wallet Electrum V2 factory class. It allows a HdWalletElectrumV2 to be created in different ways.

m_mnemonic_type: **ElectrumV2MnemonicTypes**

m_electrum_cls: **Type[ElectrumV2Base]**

CreateRandom(*wallet_name: str, words_num: ElectrumV2WordsNum = ElectrumV2WordsNum.WORDS_NUM_12, lang: ElectrumV2Languages = ElectrumV2Languages.ENGLISH*) → *HdWalletBase*

Create wallet randomly.

Parameters

- **wallet_name** (*str*) – Wallet name
- **words_num** (*HdWalletElectrumV2WordsNum, optional*) – Words number (default: 12)
- **lang** (*HdWalletElectrumV2Languages, optional*) – Language (default: English)

Returns

HdWalletBase object

Return type

HdWalletBase object

Raises

TypeError – If words number is not a HdWalletElectrumV2WordsNum enum or language is not a HdWalletElectrumV2Languages enum

CreateFromMnemonic(*wallet_name: str, mnemonic: str, passphrase: str = ""*) → *HdWalletBase*

Create wallet from mnemonic.

Parameters

- **wallet_name** (*str*) – Wallet name
- **mnemonic** (*str*) – Mnemonic
- **passphrase** (*str, optional*) – Passphrase for protecting mnemonic, empty if not specified

Returns

HdWalletBase object

Return type

HdWalletBase object

Raises

ValueError – If the mnemonic is not valid

CreateFromSeed(*wallet_name: str, seed_bytes: bytes*) → *HdWalletBase*

Create wallet from seed.

Parameters

- **wallet_name** (*str*) – Wallet name

- **seed_bytes** (*bytes*) – Seed bytes

Returns

HdWalletBase object

Return type

HdWalletBase object

Raises**ValueError** – If the seed is not valid**CreateFromExtendedKey**(*wallet_name: str, ex_key_str: str*) → *HdWalletBase*

Create wallet from extended key.

Parameters

- **wallet_name** (*str*) – Wallet name
- **ex_key_str** (*str*) – Extended key string

Returns

HdWalletBase object

Return type

HdWalletBase object

Raises**ValueError** – If the extended key is public or not valid

9.1.5.2.5 hd_wallet_electrum_v2_keys

Module with helper class for storing Electrum V2 keys.

class HdWalletElectrumV1KeyUtils

Bases: object

Class container for HD wallet Electrum keys utility functions.

static PrivToWif(*priv_key: Bip32PrivateKey*) → str

Encode private key to WIF.

Parameters**priv_key** (*Bip32PrivateKey object*) – BIP32 private key**Returns**

WIF encoded key

Return type

str

class HdWalletElectrumV2MasterKeys(*electrum_obj: ElectrumV2Base*)Bases: *HdWalletKeysBase*

HD wallet Electrum master keys class. It creates master keys from an ElectrumV2 object and store them. Keys can be got individually, as dictionary or in JSON format.

m_key_enum: *Type[Enum]***m_dict_data**: *Dict[str, Any]*

```
class HdWalletElectrumV2DerivedKeys(electrum_obj: ElectrumV2Base, change_idx: int, addr_num: int,  
                                   addr_off: int)
```

Bases: *HdWalletKeysBase*

HD wallet Electrum derived keys class. It creates derived keys from an Electrum object and store them. Keys can be got individually, as dictionary or in JSON format.

m_key_enum: `Type[Enum]`

m_dict_data: `Dict[str, Any]`

9.1.6 monero

9.1.6.1 hd_wallet_monero

Module for generating Monero wallets.

```
class HdWalletMonero(wallet_name: str, monero_obj: Monero, mnemonic: str = "", seed_bytes: bytes = b"
```

Bases: *HdWalletBase*

HD wallet Monero class. It allows to generate a Monero wallet like the official one.

m_monero_obj: `Monero`

Generate(***kwargs: Any*) → None

Generate wallet keys and addresses.

Parameters

- **acc_idx**(*int, optional*) – Account index (default: 0)
- **subaddr_num**(*int, optional*) – Subaddress number (default: 0)
- **subaddr_off**(*int, optional*) – Starting subaddress index (default: 0)

IsWatchOnly() → bool

Get if the wallet is watch-only.

Returns :

bool: True if watch-only, false otherwise

9.1.6.2 hd_wallet_monero_enum

Module with enums for Monero wallets.

```
class HdWalletMoneroDataTypes(value)
```

Bases: *HdWalletDataTypes*

Enumerative for wallet Monero data types.

WALLET_NAME = 1

COIN_NAME = 2

MNEMONIC = 3

SEED_BYTES = 4

KEY = 5

ACCOUNT_IDX = 6

SUBADDRESS_OFF = 7

SUBADDRESS = 8

class HdWalletMoneroKeyTypes(value)

Bases: [HdWalletKeyTypes](#)

Enumerative for wallet Monero key types.

PRIV_SPEND = 1

PRIV_VIEW = 2

PUB_SPEND = 3

PUB_VIEW = 4

PRIMARY_ADDRESS = 5

9.1.6.3 hd_wallet_monero_factory

Module for creating Monero wallet factories.

class HdWalletMoneroFactory(coin_type: MoneroCoins = MoneroCoins.MONERO_MAINNET)

Bases: object

HD wallet Monero factory class. It allows a HdWalletMonero to be created in different ways.

m_monero_coin: MoneroCoins

CreateRandom(wallet_name: str, words_num: MoneroWordsNum = MoneroWordsNum.WORDS_NUM_25, lang: MoneroLanguages = MoneroLanguages.ENGLISH) → [HdWalletBase](#)

Create wallet randomly.

Parameters

- **wallet_name** (str) – Wallet name
- **words_num** (HdWalletMoneroWordsNum, optional) – Words number (default: 25)
- **lang** (HdWalletMoneroLanguages, optional) – Language (default: English)

Returns

HdWalletBase object

Return type

HdWalletBase object

Raises

TypeError – If words number is not a HdWalletMoneroWordsNum enum or language is not a HdWalletMoneroLanguages enum

CreateFromMnemonic(wallet_name: str, mnemonic: str) → [HdWalletBase](#)

Create wallet from mnemonic.

Parameters

- **wallet_name** (str) – Wallet name

- **mnemonic** (*str*) – Mnemonic

Returns

HdWalletBase object

Return type

HdWalletBase object

Raises

ValueError – If the mnemonic is not valid

CreateFromSeed(*wallet_name: str, seed_bytes: bytes*) → *HdWalletBase*

Create wallet from seed.

Parameters

- **wallet_name** (*str*) – Wallet name
- **seed_bytes** (*bytes*) – Seed bytes

Returns

HdWalletBase object

Return type

HdWalletBase object

Raises

ValueError – If the seed is not valid

CreateFromPrivateKey(*wallet_name: str, priv_skey_bytes: bytes*) → *HdWalletBase*

Create wallet from private spend key.

Parameters

- **wallet_name** (*str*) – Wallet name
- **priv_skey_bytes** (*bytes*) – Private spend key bytes

Returns

HdWalletBase object

Return type

HdWalletBase object

Raises

ValueError – If the private key is not valid

CreateFromWatchOnly(*wallet_name: str, priv_vkey_bytes: bytes, pub_skey_bytes: bytes*) → *HdWalletBase*

Create wallet from private view key and public spend key (i.e. watch-only wallet).

Parameters

- **wallet_name** (*str*) – Wallet name
- **priv_vkey_bytes** (*bytes*) – Private view key bytes
- **pub_skey_bytes** (*bytes*) – Public spend key bytes

Returns

HdWalletBase object

Return type

HdWalletBase object

Raises

ValueError – If the public key is not valid

9.1.6.4 hd_wallet_monero_keys

Module with helper class for storing Monero keys.

class HdWalletMoneroKeys(*monero_obj: Monero*)

Bases: *HdWalletKeysBase*

HD wallet Monero keys class. It creates keys from a Monero object and stores them. Keys can be got individually, as dictionary or in JSON format.

m_key_enum: *Type[Enum]*

m_dict_data: *Dict[str, Any]*

9.1.6.5 hd_wallet_monero_subaddr

Module with helper class for storing Monero subaddresses.

class HdWalletMoneroSubaddressesConst

Bases: *object*

Class container for HD wallet Monero subaddresses constants.

DICT_KEY_FORMAT: *str = 'subaddress_{:d}'*

class HdWalletMoneroSubaddresses(*monero_obj: Monero, acc_idx: int, subaddr_num: int, subaddr_off: int*)

Bases: *HdWalletAddrBase*

HD wallet Monero subaddresses class. It creates subaddresses from a Monero object and stores them. Subaddresses can be got individually, as dictionary or in JSON format.

m_addr_off: *int*

m_addr: *List[Any]*

m_dict_key_str_format: *str*

9.1.7 saver

9.1.7.1 hd_wallet_saver

Module for saving wallets to file.

class HdWalletSaver(*hd_wallet: HdWalletBase*)

Bases: *object*

HD wallet saver class. It saves a wallet to file.

m_hd_wallet: *HdWalletBase*

SaveToFile(*file_path: str*) → *None*

Save wallet to file in JSON format.

Parameters

file_path (*str*) – File path

9.1.8 substrate

9.1.8.1 hd_wallet_substrate

Module for generating Substrate wallets.

```
class HdWalletSubstrate(wallet_name: str, substrate_obj: Substrate, mnemonic: str = "", passphrase: str = "",
                        seed_bytes: bytes = b")
```

Bases: *HdWalletBase*

HD wallet Substrate class. It allows to generate a Substrate wallet like the official one.

m_substrate_obj: *Substrate*

Generate(**kwargs: Any) → None

Generate wallet keys and addresses.

Parameters

path (str, optional) – Derivation path (default: empty)

IsWatchOnly() → bool

Get if the wallet is watch-only.

Returns :

bool: True if watch-only, false otherwise

9.1.8.2 hd_wallet_substrate_enum

Module with enums for Substrate wallets.

```
class HdWalletSubstrateDataTypes(value)
```

Bases: *HdWalletDataTypes*

Enumerative for wallet Substrate data types.

WALLET_NAME = 1

COIN_NAME = 2

MNEMONIC = 3

PASSPHRASE = 4

SEED_BYTES = 5

PATH = 6

KEY = 7

```
class HdWalletSubstrateKeyTypes(value)
```

Bases: *HdWalletKeyTypes*

Enumerative for wallet Substrate key types.

PRIV = 1

PUB = 2

ADDRESS = 3

9.1.8.3 hd_wallet_substrate_factory

Module for creating Substrate wallet factories.

class HdWalletSubstrateFactory(*coin_type: SubstrateCoins*)

Bases: object

HD wallet Substrate factory class. It allows a HdWalletSubstrate to be created in different ways.

m_substrate_coin: **SubstrateCoins**

CreateRandom(*wallet_name: str, words_num: Bip39WordsNum = Bip39WordsNum.WORDS_NUM_24, lang: Bip39Languages = Bip39Languages.ENGLISH*) → *HdWalletBase*

Create wallet randomly.

Parameters

- **wallet_name** (*str*) – Wallet name
- **words_num** (*HdWalletSubstrateWordsNum, optional*) – Words number (default: 24)
- **lang** (*HdWalletSubstrateLanguages, optional*) – Language (default: English)

Returns

HdWalletBase object

Return type

HdWalletBase object

Raises

TypeError – If words number is not a HdWalletSubstrateWordsNum enum or language is not a HdWalletSubstrateLanguages enum

CreateFromMnemonic(*wallet_name: str, mnemonic: str, passphrase: str = ""*) → *HdWalletBase*

Create wallet from mnemonic.

Parameters

- **wallet_name** (*str*) – Wallet name
- **mnemonic** (*str*) – Mnemonic
- **passphrase** (*str, optional*) – Passphrase for protecting mnemonic, empty if not specified

Returns

HdWalletBase object

Return type

HdWalletBase object

Raises

ValueError – If the mnemonic is not valid

CreateFromSeed(*wallet_name: str, seed_bytes: bytes*) → *HdWalletBase*

Create wallet from seed.

Parameters

- **wallet_name** (*str*) – Wallet name
- **seed_bytes** (*bytes*) – Seed bytes

Returns

HdWalletBase object

Return type

HdWalletBase object

Raises**ValueError** – If the seed is not valid**CreateFromPrivateKey**(*wallet_name: str, priv_key_bytes: bytes*) → *HdWalletBase*

Create wallet from private key.

Parameters

- **wallet_name** (*str*) – Wallet name
- **priv_key_bytes** (*bytes*) – Private key bytes

Returns

HdWalletBase object

Return type

HdWalletBase object

Raises**ValueError** – If the private key is not valid**CreateFromPublicKey**(*wallet_name: str, pub_key_bytes: bytes*) → *HdWalletBase*

Create wallet from public key.

Parameters

- **wallet_name** (*str*) – Wallet name
- **pub_key_bytes** (*bytes*) – Public key bytes

Returns

HdWalletBase object

Return type

HdWalletBase object

Raises**ValueError** – If the public key is not valid

9.1.8.4 hd_wallet_substrate_keys

Module with helper class for storing Substrate keys.

class HdWalletSubstrateKeys(*substrate_obj: Substrate*)Bases: *HdWalletKeysBase*

HD wallet Substrate keys class. It creates keys from a Substrate object and stores them. Keys can be got individually, as dictionary or in JSON format.

m_key_enum: Type[Enum]**m_dict_data:** Dict[str, Any]

9.1.9 utils

9.1.9.1 utils

Module with some utility functions.

class Utils

Bases: object

Class container for utility functions.

static BytesToHexString(*data_bytes: bytes, encoding: str = 'utf-8'*) → str

Convert bytes to hex string.

Parameters

- **data_bytes** (*bytes*) – Data bytes
- **encoding** (*str, optional*) – Encoding type

Returns

Bytes converted to hex string

Return type

str

PYTHON MODULE INDEX

[p](#)
[py_crypto_hd_wallet.algorand.hd_wallet_algorand](#), [33](#)
[17](#)
[py_crypto_hd_wallet.algorand.hd_wallet_algorand_enum](#), [34](#)
[17](#)
[py_crypto_hd_wallet.algorand.hd_wallet_algorand_factory](#), [35](#)
[18](#)
[py_crypto_hd_wallet.algorand.hd_wallet_algorand_keys](#), [36](#)
[20](#)
[py_crypto_hd_wallet.bip.hd_wallet_bip](#), [20](#)
[py_crypto_hd_wallet.bip.hd_wallet_bip_addr](#), [37](#)
[20](#)
[py_crypto_hd_wallet.bip.hd_wallet_bip_enum](#), [37](#)
[21](#)
[py_crypto_hd_wallet.bip.hd_wallet_bip_factory](#), [38](#)
[22](#)
[py_crypto_hd_wallet.bip.hd_wallet_bip_keys](#), [39](#)
[23](#)
[py_crypto_hd_wallet.cardano.shelley.hd_wallet_cardano_shelley](#), [40](#)
[24](#)
[py_crypto_hd_wallet.cardano.shelley.hd_wallet_cardano_shelley_addr](#), [41](#)
[24](#)
[py_crypto_hd_wallet.cardano.shelley.hd_wallet_cardano_shelley_enum](#), [42](#)
[25](#)
[py_crypto_hd_wallet.cardano.shelley.hd_wallet_cardano_shelley_factory](#), [43](#)
[25](#)
[py_crypto_hd_wallet.cardano.shelley.hd_wallet_cardano_shelley_keys](#), [43](#)
[27](#)
[py_crypto_hd_wallet.common.hd_wallet_addr_base](#), [44](#)
[28](#)
[py_crypto_hd_wallet.common.hd_wallet_base](#), [29](#)
[py_crypto_hd_wallet.common.hd_wallet_data_types](#), [45](#)
[30](#)
[py_crypto_hd_wallet.common.hd_wallet_enum_dict](#), [46](#)
[31](#)
[py_crypto_hd_wallet.common.hd_wallet_keys_base](#), [47](#)
[31](#)
[py_crypto_hd_wallet.electrum.v1.hd_wallet_electrum_v1](#),
[32](#)
[py_crypto_hd_wallet.electrum.v1.hd_wallet_electrum_v1_addr](#),
[33](#)
[py_crypto_hd_wallet.electrum.v1.hd_wallet_electrum_v1_enum](#),

Symbols

`__getitem__()` (*HdWalletAddrBase* method), 29
`__iter__()` (*HdWalletAddrBase* method), 29

A

`ACCOUNT_IDX` (*HdWalletBipDataTypes* attribute), 21
`ACCOUNT_IDX` (*HdWalletCardanoShelleyDataTypes* attribute), 25
`ACCOUNT_IDX` (*HdWalletMoneroDataTypes* attribute), 41
`ACCOUNT_KEY` (*HdWalletBipDataTypes* attribute), 21
`ACCOUNT_KEY` (*HdWalletCardanoShelleyDataTypes* attribute), 25
`ADDRESS` (*HdWalletAlgorandKeyTypes* attribute), 18
`ADDRESS` (*HdWalletBipDataTypes* attribute), 21
`ADDRESS` (*HdWalletBipKeyTypes* attribute), 21
`ADDRESS` (*HdWalletCardanoShelleyDataTypes* attribute), 25
`ADDRESS` (*HdWalletCardanoShelleyKeyTypes* attribute), 25
`ADDRESS` (*HdWalletElectrumV1DataTypes* attribute), 33
`ADDRESS` (*HdWalletElectrumV1KeyTypes* attribute), 33
`ADDRESS` (*HdWalletElectrumV2DataTypes* attribute), 37
`ADDRESS` (*HdWalletElectrumV2KeyTypes* attribute), 37
`ADDRESS` (*HdWalletSubstrateKeyTypes* attribute), 44
`ADDRESS_OFF` (*HdWalletBipDataTypes* attribute), 21
`ADDRESS_OFF` (*HdWalletCardanoShelleyDataTypes* attribute), 25
`ADDRESS_OFF` (*HdWalletElectrumV1DataTypes* attribute), 33
`ADDRESS_OFF` (*HdWalletElectrumV2DataTypes* attribute), 37

B

`BIP_COIN_TO_CLASS` (*HdWalletBipFactoryConst* attribute), 22
`BytesToHexString()` (*Utils* static method), 47

C

`CHANGE_IDX` (*HdWalletBipDataTypes* attribute), 21
`CHANGE_IDX` (*HdWalletCardanoShelleyDataTypes* attribute), 25

`CHANGE_IDX` (*HdWalletElectrumV1DataTypes* attribute), 33
`CHANGE_IDX` (*HdWalletElectrumV2DataTypes* attribute), 37
`CHANGE_KEY` (*HdWalletBipDataTypes* attribute), 21
`COIN_KEY` (*HdWalletBipDataTypes* attribute), 21
`COIN_NAME` (*HdWalletAlgorandDataTypes* attribute), 17
`COIN_NAME` (*HdWalletBipDataTypes* attribute), 21
`COIN_NAME` (*HdWalletCardanoShelleyDataTypes* attribute), 25
`COIN_NAME` (*HdWalletElectrumV1DataTypes* attribute), 33
`COIN_NAME` (*HdWalletElectrumV2DataTypes* attribute), 37
`COIN_NAME` (*HdWalletMoneroDataTypes* attribute), 40
`COIN_NAME` (*HdWalletSubstrateDataTypes* attribute), 44
`Count()` (*HdWalletAddrBase* method), 29
`CreateFromExtendedKey()` (*HdWalletBipFactory* method), 23
`CreateFromExtendedKey()` (*HdWalletElectrumV2Factory* method), 39
`CreateFromMnemonic()` (*HdWalletAlgorandFactory* method), 18
`CreateFromMnemonic()` (*HdWalletBipFactory* method), 22
`CreateFromMnemonic()` (*HdWalletCardanoShelleyFactory* method), 26
`CreateFromMnemonic()` (*HdWalletElectrumV1Factory* method), 34
`CreateFromMnemonic()` (*HdWalletElectrumV2Factory* method), 38
`CreateFromMnemonic()` (*HdWalletMoneroFactory* method), 41
`CreateFromMnemonic()` (*HdWalletSubstrateFactory* method), 45
`CreateFromPrivateKey()` (*HdWalletAlgorandFactory* method), 19
`CreateFromPrivateKey()` (*HdWalletCardanoShelleyFactory* method), 26
`CreateFromPrivateKey()` (*HdWalletElectrumV1Factory* method), 35
`CreateFromPrivateKey()` (*HdWalletMoneroFactory* method), 41

method), 42
CreateFromPrivateKey() (HdWalletSubstrateFactory method), 46
CreateFromPublicKey() (HdWalletAlgorandFactory method), 19
CreateFromPublicKey() (HdWalletCardanoShelleyFactory method), 27
CreateFromPublicKey() (HdWalletElectrumV1Factory method), 35
CreateFromPublicKey() (HdWalletSubstrateFactory method), 46
CreateFromSeed() (HdWalletAlgorandFactory method), 18
CreateFromSeed() (HdWalletBipFactory method), 23
CreateFromSeed() (HdWalletCardanoShelleyFactory method), 26
CreateFromSeed() (HdWalletElectrumV1Factory method), 34
CreateFromSeed() (HdWalletElectrumV2Factory method), 38
CreateFromSeed() (HdWalletMoneroFactory method), 42
CreateFromSeed() (HdWalletSubstrateFactory method), 45
CreateFromWatchOnly() (HdWalletMoneroFactory method), 42
CreateRandom() (HdWalletAlgorandFactory method), 18
CreateRandom() (HdWalletBipFactory method), 22
CreateRandom() (HdWalletCardanoShelleyFactory method), 25
CreateRandom() (HdWalletElectrumV1Factory method), 34
CreateRandom() (HdWalletElectrumV2Factory method), 38
CreateRandom() (HdWalletMoneroFactory method), 41
CreateRandom() (HdWalletSubstrateFactory method), 45

D

DICT_KEY_DEF_FORMAT (HdWalletAddrBaseConst attribute), 28
DICT_KEY_FORMAT (HdWalletMoneroSubaddressesConst attribute), 43

E

EX_PRIV (HdWalletBipKeyTypes attribute), 21
EX_PRIV (HdWalletElectrumV2KeyTypes attribute), 37
EX_PUB (HdWalletBipKeyTypes attribute), 21
EX_PUB (HdWalletElectrumV2KeyTypes attribute), 37

G

Generate() (HdWalletAlgorand method), 17
Generate() (HdWalletBase method), 29

Generate() (HdWalletBip method), 20
Generate() (HdWalletCardanoShelley method), 24
Generate() (HdWalletElectrumV1 method), 32
Generate() (HdWalletElectrumV2 method), 36
Generate() (HdWalletMonero method), 40
Generate() (HdWalletSubstrate method), 44
GetData() (HdWalletBase method), 30
GetKey() (HdWalletKeysBase method), 32

H

HasData() (HdWalletBase method), 30
HasKey() (HdWalletKeysBase method), 31
HdWalletAddrBase (class in py_crypto_hd_wallet.common.hd_wallet_addr_base), 28
HdWalletAddrBaseConst (class in py_crypto_hd_wallet.common.hd_wallet_addr_base), 28
HdWalletAlgorand (class in py_crypto_hd_wallet.algorand.hd_wallet_algorand), 17
HdWalletAlgorandDataTypes (class in py_crypto_hd_wallet.algorand.hd_wallet_algorand_enum), 17
HdWalletAlgorandFactory (class in py_crypto_hd_wallet.algorand.hd_wallet_algorand_factory), 18
HdWalletAlgorandKeys (class in py_crypto_hd_wallet.algorand.hd_wallet_algorand_keys), 20
HdWalletAlgorandKeyTypes (class in py_crypto_hd_wallet.algorand.hd_wallet_algorand_enum), 17
HdWalletBase (class in py_crypto_hd_wallet.common.hd_wallet_base), 29
HdWalletBip (class in py_crypto_hd_wallet.bip.hd_wallet_bip), 20
HdWalletBipAddresses (class in py_crypto_hd_wallet.bip.hd_wallet_bip_addr), 20
HdWalletBipDataTypes (class in py_crypto_hd_wallet.bip.hd_wallet_bip_enum), 21
HdWalletBipFactory (class in py_crypto_hd_wallet.bip.hd_wallet_bip_factory), 22
HdWalletBipFactoryConst (class in py_crypto_hd_wallet.bip.hd_wallet_bip_factory), 22
HdWalletBipKeys (class in py_crypto_hd_wallet.bip.hd_wallet_bip_keys), 23

HdWalletBipKeyTypes	(class in HdWalletElectrumV1KeyUtils	(class in
py_crypto_hd_wallet.bip.hd_wallet_bip_enum),	py_crypto_hd_wallet.electrum.v2.hd_wallet_electrum_v2_keys),	
21	39	
HdWalletCardanoShelley	(class in HdWalletElectrumV1MasterKeys	(class in
py_crypto_hd_wallet.cardano.shelley.hd_wallet_cardano_shelley),	py_crypto_hd_wallet.electrum.v1.hd_wallet_electrum_v1_keys),	
24	35	
HdWalletCardanoShelleyAddresses	(class in HdWalletElectrumV2	(class in
py_crypto_hd_wallet.cardano.shelley.hd_wallet_cardano_shelley_addresses),	py_crypto_hd_wallet.electrum.v2.hd_wallet_electrum_v2),	
24	36	
HdWalletCardanoShelleyDataTypes	(class in HdWalletElectrumV2Addresses	(class in
py_crypto_hd_wallet.cardano.shelley.hd_wallet_cardano_shelley_data_types),	py_crypto_hd_wallet.electrum.v2.hd_wallet_electrum_v2_addr),	
25	37	
HdWalletCardanoShelleyDerivedKeys	(class in HdWalletElectrumV2DataTypes	(class in
py_crypto_hd_wallet.cardano.shelley.hd_wallet_cardano_shelley_derived_keys),	py_crypto_hd_wallet.electrum.v2.hd_wallet_electrum_v2_enum),	
28	37	
HdWalletCardanoShelleyFactory	(class in HdWalletElectrumV2DerivedKeys	(class in
py_crypto_hd_wallet.cardano.shelley.hd_wallet_cardano_shelley_factory),	py_crypto_hd_wallet.electrum.v2.hd_wallet_electrum_v2_keys),	
25	39	
HdWalletCardanoShelleyKeysBase	(class in HdWalletElectrumV2Factory	(class in
py_crypto_hd_wallet.cardano.shelley.hd_wallet_cardano_shelley_keys_base),	py_crypto_hd_wallet.electrum.v2.hd_wallet_electrum_v2_factory	
27	38	
HdWalletCardanoShelleyKeyTypes	(class in HdWalletElectrumV2KeyTypes	(class in
py_crypto_hd_wallet.cardano.shelley.hd_wallet_cardano_shelley_key_types),	py_crypto_hd_wallet.electrum.v2.hd_wallet_electrum_v2_enum),	
25	37	
HdWalletCardanoShelleyMasterKeys	(class in HdWalletElectrumV2MasterKeys	(class in
py_crypto_hd_wallet.cardano.shelley.hd_wallet_cardano_shelley_master_keys),	py_crypto_hd_wallet.electrum.v2.hd_wallet_electrum_v2_keys),	
27	39	
HdWalletCardanoShelleyStakingKeys	(class in HdWalletEnumDict	(class in
py_crypto_hd_wallet.cardano.shelley.hd_wallet_cardano_shelley_staking_keys),	py_crypto_hd_wallet.common.hd_wallet_enum_dict),	
27	31	
HdWalletDataTypes	(class in HdWalletKeysBase	(class in
py_crypto_hd_wallet.common.hd_wallet_data_types),	py_crypto_hd_wallet.common.hd_wallet_keys_base),	
30	31	
HdWalletElectrumV1	(class in HdWalletKeyTypes	(class in
py_crypto_hd_wallet.electrum.v1.hd_wallet_electrum_v1),	py_crypto_hd_wallet.common.hd_wallet_data_types),	
32	30	
HdWalletElectrumV1Addresses	(class in HdWalletMonero	(class in
py_crypto_hd_wallet.electrum.v1.hd_wallet_electrum_v1_addresses),	py_crypto_hd_wallet.monero.hd_wallet_monero),	
33	40	
HdWalletElectrumV1DataTypes	(class in HdWalletMoneroDataTypes	(class in
py_crypto_hd_wallet.electrum.v1.hd_wallet_electrum_v1_data_types),	py_crypto_hd_wallet.monero.hd_wallet_monero_enum),	
33	40	
HdWalletElectrumV1DerivedKeys	(class in HdWalletMoneroFactory	(class in
py_crypto_hd_wallet.electrum.v1.hd_wallet_electrum_v1_derived_keys),	py_crypto_hd_wallet.monero.hd_wallet_monero_factory),	
36	41	
HdWalletElectrumV1Factory	(class in HdWalletMoneroKeys	(class in
py_crypto_hd_wallet.electrum.v1.hd_wallet_electrum_v1_factory),	py_crypto_hd_wallet.monero.hd_wallet_monero_keys),	
34	43	
HdWalletElectrumV1KeyTypes	(class in HdWalletMoneroKeyTypes	(class in
py_crypto_hd_wallet.electrum.v1.hd_wallet_electrum_v1_key_types),	py_crypto_hd_wallet.monero.hd_wallet_monero_enum),	
33	41	
HdWalletElectrumV1KeyUtils	(class in HdWalletMoneroSubaddresses	(class in
py_crypto_hd_wallet.electrum.v1.hd_wallet_electrum_v1_key_utils),	py_crypto_hd_wallet.monero.hd_wallet_monero_subaddr),	
35	43	

HdWalletMoneroSubaddressesConst	(class in	m_addr_off (HdWalletMoneroSubaddresses attribute),
py_crypto_hd_wallet.monero.hd_wallet_monero_subaddr),	43	
		m_bip_cls (HdWalletBipFactory attribute), 22
HdWalletSaver	(class in	m_bip_coin (HdWalletBipFactory attribute), 22
py_crypto_hd_wallet.saver.hd_wallet_saver),	43	m_bip_obj (HdWalletAlgorand attribute), 17
		m_bip_obj (HdWalletBip attribute), 20
HdWalletSubstrate	(class in	m_bip_obj (HdWalletCardanoShelley attribute), 24
py_crypto_hd_wallet.substrate.hd_wallet_substrate),	44	m_coin (HdWalletCardanoShelleyFactory attribute), 25
		m_dict_data (HdWalletAlgorandKeys attribute), 20
HdWalletSubstrateDataTypes	(class in	m_dict_data (HdWalletBase attribute), 30
py_crypto_hd_wallet.substrate.hd_wallet_substrate_data_types),	44	m_dict_data (HdWalletBipKeys attribute), 23
		m_dict_data (HdWalletCardanoShelleyDerivedKeys attribute), 28
HdWalletSubstrateFactory	(class in	m_dict_data (HdWalletCardanoShelleyKeysBase attribute), 27
py_crypto_hd_wallet.substrate.hd_wallet_substrate_factory),	45	
HdWalletSubstrateKeys	(class in	m_dict_data (HdWalletCardanoShelleyMasterKeys attribute), 27
py_crypto_hd_wallet.substrate.hd_wallet_substrate_keys),	46	m_dict_data (HdWalletCardanoShelleyStakingKeys attribute), 28
HdWalletSubstrateKeyTypes	(class in	m_dict_data (HdWalletElectrumV1DerivedKeys attribute), 36
py_crypto_hd_wallet.substrate.hd_wallet_substrate_key_types),	44	m_dict_data (HdWalletElectrumV1MasterKeys attribute), 36
		m_dict_data (HdWalletElectrumV2DerivedKeys attribute), 40
		m_dict_data (HdWalletElectrumV2MasterKeys attribute), 39
		m_dict_data (HdWalletEnumDict attribute), 31
		m_dict_data (HdWalletKeysBase attribute), 32
		m_dict_data (HdWalletMoneroKeys attribute), 43
		m_dict_data (HdWalletSubstrateKeys attribute), 46
		m_dict_key_str_format (HdWalletAddrBase attribute), 28
		m_dict_key_str_format (HdWalletBipAddresses attribute), 21
		m_dict_key_str_format (HdWalletCardanoShelleyAddresses attribute), 24
		m_dict_key_str_format (HdWalletElectrumV1Addresses attribute), 33
		m_dict_key_str_format (HdWalletElectrumV2Addresses attribute), 37
		m_dict_key_str_format (HdWalletMoneroSubaddresses attribute), 43
		m_electrum_cls (HdWalletElectrumV2Factory attribute), 38
		m_electrum_obj (HdWalletElectrumV1 attribute), 32
		m_electrum_obj (HdWalletElectrumV2 attribute), 36
		m_hd_wallet (HdWalletSaver attribute), 43
		m_key_enum (HdWalletAlgorandKeys attribute), 20
		m_key_enum (HdWalletBase attribute), 30
		m_key_enum (HdWalletBipKeys attribute), 23
		m_key_enum (HdWalletCardanoShelleyDerivedKeys attribute), 28

m_key_enum (HdWalletCardanoShelleyKeysBase attribute), 27	py_crypto_hd_wallet.bip.hd_wallet_bip_factory, 22
m_key_enum (HdWalletCardanoShelleyMasterKeys attribute), 27	py_crypto_hd_wallet.bip.hd_wallet_bip_keys, 23
m_key_enum (HdWalletCardanoShelleyStakingKeys attribute), 28	py_crypto_hd_wallet.cardano.shelley.hd_wallet_cardano, 24
m_key_enum (HdWalletElectrumV1DerivedKeys attribute), 36	py_crypto_hd_wallet.cardano.shelley.hd_wallet_cardano, 24
m_key_enum (HdWalletElectrumV1MasterKeys attribute), 36	py_crypto_hd_wallet.cardano.shelley.hd_wallet_cardano, 25
m_key_enum (HdWalletElectrumV2DerivedKeys attribute), 40	py_crypto_hd_wallet.cardano.shelley.hd_wallet_cardano, 25
m_key_enum (HdWalletElectrumV2MasterKeys attribute), 39	py_crypto_hd_wallet.cardano.shelley.hd_wallet_cardano, 27
m_key_enum (HdWalletEnumDict attribute), 31	py_crypto_hd_wallet.common.hd_wallet_addr_base, 28
m_key_enum (HdWalletKeysBase attribute), 32	py_crypto_hd_wallet.common.hd_wallet_base, 29
m_key_enum (HdWalletMoneroKeys attribute), 43	py_crypto_hd_wallet.common.hd_wallet_data_types, 30
m_key_enum (HdWalletSubstrateKeys attribute), 46	py_crypto_hd_wallet.common.hd_wallet_enum_dict, 31
m_mnemonic_type (HdWalletElectrumV2Factory attribute), 38	py_crypto_hd_wallet.common.hd_wallet_keys_base, 31
m_monero_coin (HdWalletMoneroFactory attribute), 41	py_crypto_hd_wallet.electrum.v1.hd_wallet_electrum_v1, 32
m_monero_obj (HdWalletMonero attribute), 40	py_crypto_hd_wallet.electrum.v1.hd_wallet_electrum_v1, 33
m_substrate_coin (HdWalletSubstrateFactory attribute), 45	py_crypto_hd_wallet.electrum.v1.hd_wallet_electrum_v1, 33
m_substrate_obj (HdWalletSubstrate attribute), 44	py_crypto_hd_wallet.electrum.v1.hd_wallet_electrum_v1, 34
MASTER_KEY (HdWalletBipDataTypes attribute), 21	py_crypto_hd_wallet.electrum.v1.hd_wallet_electrum_v1, 35
MASTER_KEY (HdWalletCardanoShelleyDataTypes attribute), 25	py_crypto_hd_wallet.electrum.v2.hd_wallet_electrum_v2, 36
MASTER_KEY (HdWalletElectrumV1DataTypes attribute), 33	py_crypto_hd_wallet.electrum.v2.hd_wallet_electrum_v2, 37
MASTER_KEY (HdWalletElectrumV2DataTypes attribute), 37	py_crypto_hd_wallet.electrum.v2.hd_wallet_electrum_v2, 37
MNEMONIC (HdWalletAlgorandDataTypes attribute), 17	py_crypto_hd_wallet.electrum.v2.hd_wallet_electrum_v2, 38
MNEMONIC (HdWalletBipDataTypes attribute), 21	py_crypto_hd_wallet.electrum.v2.hd_wallet_electrum_v2, 39
MNEMONIC (HdWalletCardanoShelleyDataTypes attribute), 25	py_crypto_hd_wallet.monero.hd_wallet_monero, 40
MNEMONIC (HdWalletElectrumV1DataTypes attribute), 33	py_crypto_hd_wallet.monero.hd_wallet_monero_enum, 41
MNEMONIC (HdWalletElectrumV2DataTypes attribute), 37	py_crypto_hd_wallet.monero.hd_wallet_monero_factory, 43
MNEMONIC (HdWalletMoneroDataTypes attribute), 40	py_crypto_hd_wallet.monero.hd_wallet_monero_keys, 43
MNEMONIC (HdWalletSubstrateDataTypes attribute), 44	py_crypto_hd_wallet.monero.hd_wallet_monero_subaddr, 43
module	
py_crypto_hd_wallet.algorand.hd_wallet_algorand, 17	
py_crypto_hd_wallet.algorand.hd_wallet_algorand_enum, 17	
py_crypto_hd_wallet.algorand.hd_wallet_algorand_factory, 18	
py_crypto_hd_wallet.algorand.hd_wallet_algorand_keys, 20	
py_crypto_hd_wallet.bip.hd_wallet_bip, 20	
py_crypto_hd_wallet.bip.hd_wallet_bip_addr, 20	
py_crypto_hd_wallet.bip.hd_wallet_bip_enum, 21	

py_crypto_hd_wallet.saver.hd_wallet_saver,	module, 24
43	py_crypto_hd_wallet.cardano.shelley.hd_wallet_cardano_shelley
py_crypto_hd_wallet.substrate.hd_wallet_substrate,	module, 25
44	py_crypto_hd_wallet.cardano.shelley.hd_wallet_cardano_shelley
py_crypto_hd_wallet.substrate.hd_wallet_substrate_enum,	module, 26
44	py_crypto_hd_wallet.cardano.shelley.hd_wallet_cardano_shelley
py_crypto_hd_wallet.substrate.hd_wallet_substrate_factory,	module, 27
45	py_crypto_hd_wallet.common.hd_wallet_addr_base
py_crypto_hd_wallet.substrate.hd_wallet_substrate_key,	module, 28
46	py_crypto_hd_wallet.common.hd_wallet_base
py_crypto_hd_wallet.utils.utils,	47 module, 29
P	py_crypto_hd_wallet.common.hd_wallet_data_types
PASSPHRASE (<i>HdWalletBipDataTypes</i> attribute),	21 module, 30
PASSPHRASE (<i>HdWalletCardanoShelleyDataTypes</i>	module, 31
attribute),	25 py_crypto_hd_wallet.common.hd_wallet_keys_base
PASSPHRASE (<i>HdWalletElectrumV2DataTypes</i> attribute),	module, 31
37	py_crypto_hd_wallet.electrum.v1.hd_wallet_electrum_v1
PASSPHRASE (<i>HdWalletSubstrateDataTypes</i> attribute),	44 module, 32
PATH (<i>HdWalletSubstrateDataTypes</i> attribute),	44 py_crypto_hd_wallet.electrum.v1.hd_wallet_electrum_v1_addr
PRIMARY_ADDRESS (<i>HdWalletMoneroKeyTypes</i> at-	module, 33
tribute),	41 py_crypto_hd_wallet.electrum.v1.hd_wallet_electrum_v1_enum
PRIV (<i>HdWalletAlgorandKeyTypes</i> attribute),	18 module, 33
PRIV (<i>HdWalletSubstrateKeyTypes</i> attribute),	44 py_crypto_hd_wallet.electrum.v1.hd_wallet_electrum_v1_fact
PRIV_SPEND (<i>HdWalletMoneroKeyTypes</i> attribute),	41 module, 34
PRIV_VIEW (<i>HdWalletMoneroKeyTypes</i> attribute),	41 py_crypto_hd_wallet.electrum.v1.hd_wallet_electrum_v1_keys
PrivToWif() (<i>HdWalletElectrumV1KeyUtils</i> static	method), 35, 39 module, 35
PUB (<i>HdWalletAlgorandKeyTypes</i> attribute),	18 py_crypto_hd_wallet.electrum.v2.hd_wallet_electrum_v2
PUB (<i>HdWalletSubstrateKeyTypes</i> attribute),	44 module, 36
PUB_SPEND (<i>HdWalletMoneroKeyTypes</i> attribute),	41 py_crypto_hd_wallet.electrum.v2.hd_wallet_electrum_v2_addr
PUB_VIEW (<i>HdWalletMoneroKeyTypes</i> attribute),	41 module, 37
PURPOSE_KEY (<i>HdWalletBipDataTypes</i> attribute),	21 py_crypto_hd_wallet.electrum.v2.hd_wallet_electrum_v2_enum
py_crypto_hd_wallet.algorand.hd_wallet_algorand,	17 module, 37
py_crypto_hd_wallet.algorand.hd_wallet_algorand_enum,	17 py_crypto_hd_wallet.electrum.v2.hd_wallet_electrum_v2_fact
py_crypto_hd_wallet.algorand.hd_wallet_algorand_factory,	18 module, 38
py_crypto_hd_wallet.algorand.hd_wallet_algorand_keys,	20 py_crypto_hd_wallet.electrum.v2.hd_wallet_electrum_v2_keys
py_crypto_hd_wallet.bip.hd_wallet_bip	20 module, 39
py_crypto_hd_wallet.bip.hd_wallet_bip_addr	20 py_crypto_hd_wallet.monero.hd_wallet_monero
py_crypto_hd_wallet.bip.hd_wallet_bip_enum	21 module, 40
py_crypto_hd_wallet.bip.hd_wallet_bip_factory	22 py_crypto_hd_wallet.monero.hd_wallet_monero_enum
py_crypto_hd_wallet.bip.hd_wallet_bip_keys	23 module, 40
py_crypto_hd_wallet.cardano.shelley.hd_wallet_cardano_shelley	24 py_crypto_hd_wallet.monero.hd_wallet_monero_factory
py_crypto_hd_wallet.cardano.shelley.hd_wallet_cardano_shelley_addr	24 module, 41
	py_crypto_hd_wallet.monero.hd_wallet_monero_keys
	module, 43
	py_crypto_hd_wallet.monero.hd_wallet_monero_subaddr
	module, 43
	py_crypto_hd_wallet.saver.hd_wallet_saver
	module, 43
	py_crypto_hd_wallet.substrate.hd_wallet_substrate
	module, 44
	py_crypto_hd_wallet.substrate.hd_wallet_substrate_enum
	module, 44
	py_crypto_hd_wallet.substrate.hd_wallet_substrate_factory

module, 45
 py_crypto_hd_wallet.substrate.hd_wallet_substrate_key\$tribute), 25
 module, 46
 py_crypto_hd_wallet.utils.utils
 module, 47

R

RAW_COMPR_PUB (*HdWalletBipKeyTypes attribute*), 21
 RAW_PRIV (*HdWalletBipKeyTypes attribute*), 21
 RAW_PRIV (*HdWalletCardanoShelleyKeyTypes attribute*), 25
 RAW_PRIV (*HdWalletElectrumV1KeyTypes attribute*), 33
 RAW_PRIV (*HdWalletElectrumV2KeyTypes attribute*), 37
 RAW_PUB (*HdWalletCardanoShelleyKeyTypes attribute*), 25
 RAW_PUB (*HdWalletElectrumV1KeyTypes attribute*), 33
 RAW_PUB (*HdWalletElectrumV2KeyTypes attribute*), 37
 RAW_UNCOMPR_PUB (*HdWalletBipKeyTypes attribute*), 21

S

SaveToFile() (*HdWalletSaver method*), 43
 SEED_BYTES (*HdWalletAlgorandDataTypes attribute*), 17
 SEED_BYTES (*HdWalletBipDataTypes attribute*), 21
 SEED_BYTES (*HdWalletCardanoShelleyDataTypes attribute*), 25
 SEED_BYTES (*HdWalletElectrumV1DataTypes attribute*), 33
 SEED_BYTES (*HdWalletElectrumV2DataTypes attribute*), 37
 SEED_BYTES (*HdWalletMoneroDataTypes attribute*), 40
 SEED_BYTES (*HdWalletSubstrateDataTypes attribute*), 44
 SPEC_NAME (*HdWalletBipDataTypes attribute*), 21
 STAKING_KEY (*HdWalletCardanoShelleyDataTypes attribute*), 25
 SUBADDRESS (*HdWalletMoneroDataTypes attribute*), 41
 SUBADDRESS_OFF (*HdWalletMoneroDataTypes attribute*), 41

T

ToDict() (*HdWalletAddrBase method*), 28
 ToDict() (*HdWalletBase method*), 29
 ToDict() (*HdWalletEnumDict method*), 31
 ToJson() (*HdWalletAddrBase method*), 28
 ToJson() (*HdWalletEnumDict method*), 31

U

Utils (*class in py_crypto_hd_wallet.utils.utils*), 47

W

WALLET_NAME (*HdWalletAlgorandDataTypes attribute*), 17
 WALLET_NAME (*HdWalletBipDataTypes attribute*), 21

WALLET_NAME (*HdWalletCardanoShelleyDataTypes attribute*), 25
 WALLET_NAME (*HdWalletElectrumV1DataTypes attribute*), 33
 WALLET_NAME (*HdWalletElectrumV2DataTypes attribute*), 37
 WALLET_NAME (*HdWalletMoneroDataTypes attribute*), 40
 WALLET_NAME (*HdWalletSubstrateDataTypes attribute*), 44
 WIF_PRIV (*HdWalletBipKeyTypes attribute*), 21
 WIF_PRIV (*HdWalletElectrumV1KeyTypes attribute*), 33
 WIF_PRIV (*HdWalletElectrumV2KeyTypes attribute*), 37