
PyXDC

Release 0.1.1

Meheret Tesfaye

May 05, 2021

CONTENTS

1	PyXDC	1
2	Installing PyXDC	3
2.1	Development	3
3	Wallet	5
4	RPC	15
5	Transaction	17
5.1	ContractTransaction	18
5.2	NormalTransaction	19
6	Signature	21
7	Utils	23
	Python Module Index	27
	Index	29

Python library with tools for XinFin blockchain.

INSTALLING PYXDC

The easiest way to install *pyxdc* is via pip:

```
$ pip install pyxdc
```

If you want to run the latest version of the code, you can install from git:

```
$ pip install git+git://github.com/meherett/pyxdc.git
```

For the versions available, see the [tags on this repository](#).

2.1 Development

We welcome pull requests. To get started, just fork this [github repository](#), clone it locally, and run:

```
$ pip install -e . -r requirements.txt
```


WALLET

```
class pyxdc.wallet.Wallet (provider: Union[web3.providers.rpc.HTTPProvider,  
                                         web3.providers.websocket.WebsocketProvider] =  
                                         <web3.providers.rpc.HTTPProvider object>, use_default_path:  
                                         bool = False)
```

XinFin Wallet.

Parameters

- **provider** (*HTTPProvider*, *WebsocketProvider*) – XinFin provider, default to HTTP_PROVIDER.
- **use_default_path** (*bool*) – Use default derivation path, defaults to False.

Returns Wallet – Wallet instance.

```
from_entropy (entropy: str, language: str = 'english', passphrase: Optional[str] = None) →  
               pyxdc.wallet.Wallet  
Master from Entropy hex string.
```

Parameters

- **entropy** (*str*) – Entropy hex string.
- **language** (*str*) – Mnemonic language, default to english.
- **passphrase** (*str*) – Mnemonic passphrase or password, default to None.

Returns Wallet – Wallet instance.

```
>>> from pyxdc import WEBSOCKET_PROVIDER  
>>> from pyxdc.wallet import Wallet  
>>> wallet: Wallet = Wallet(provider=WEBSOCKET_PROVIDER)  
>>> wallet.from_entropy(entropy="b64dc1c3c3d5b876a94006d49c1e4ed2f106b86e",  
↳ language="english", passphrase=None)  
<pyxdc.wallet.Wallet object at 0x000001E8BFB98D60>
```

```
from_mnemonic (mnemonic: str, language: Optional[str] = None, passphrase: Optional[str] = None)  
               → pyxdc.wallet.Wallet  
Master from Mnemonic words.
```

Parameters

- **mnemonic** (*str*) – Mnemonic words.
- **language** (*str*) – Mnemonic language, default to None.
- **passphrase** (*str*) – Mnemonic passphrase or password, default to None.

Returns Wallet – Wallet instance.

```
>>> from pyxdc import HTTP_PROVIDER
>>> from pyxdc.wallet import Wallet
>>> wallet: Wallet = Wallet(provider=HTTP_PROVIDER)
>>> wallet.from_mnemonic(mnemonic="rent host ill marble fortune deputy pink_
↳ absorb stand thought neck planet away found robust", passphrase=None)
<pyxdc.wallet.Wallet object at 0x000001E8BFB98D60>
```

from_seed (*seed: str*) → *pyxdc.wallet.Wallet*

Master from Seed hex string.

Parameters *seed* (*str*) – Seed hex string.

Returns *Wallet* – Wallet instance.

```
>>> from pyxdc import HTTP_PROVIDER
>>> from pyxdc.wallet import Wallet
>>> wallet: Wallet = Wallet(provider=HTTP_PROVIDER)
>>> wallet.from_seed(seed=
↳ "09d6f96646d69b3842eecb8f05737972c6c0314d60c203657ae2dad5e8dd88797019ad9938292307de2f4a740
↳ ")
<pyxdc.wallet.Wallet object at 0x000001E8BFB98D60>
```

from_root_xprivate_key (*root_xprivate_key: str*) → *pyxdc.wallet.Wallet*

Master from Root XPrivate Key.

Parameters *root_xprivate_key* (*str*) – Root xprivate key.

Returns *Wallet* – Wallet instance.

```
>>> from pyxdc import HTTP_PROVIDER
>>> from pyxdc.wallet import Wallet
>>> wallet: Wallet = Wallet(provider=HTTP_PROVIDER)
>>> wallet.from_root_xprivate_key(root_xprivate_key=
↳ "xprv9s21ZrQH143K3i9qWtFiAawwn2iLAcKKfXHCsTdUsy7RYSama9qzrrwEwsu9buLocH7qFQmTow5bSysKDmq8V
↳ ")
<pyxdc.wallet.Wallet object at 0x000001E8BFB98D60>
```

from_xprivate_key (*xprivate_key: str*) → *pyxdc.wallet.Wallet*

Master from XPrivate Key.

Parameters *xprivate_key* (*str*) – XPrivate key.

Returns *Wallet* – Wallet instance.

```
>>> from pyxdc import HTTP_PROVIDER
>>> from pyxdc.wallet import Wallet
>>> wallet: Wallet = Wallet(provider=HTTP_PROVIDER)
>>> wallet.from_xprivate_key(xprivate_key=
↳ "xprvA2oDuneWodkNiecDi8VoBCvu7TSfnDmGqr5oKzkkLWvmE9dm1TQzYcp9HZQLqYTep1T3yykxZgiUSJDZYrvnr
↳ ")
<pyxdc.wallet.Wallet object at 0x000001E8BFB98D60>
```

from_wif (*wif: str*) → *pyxdc.wallet.Wallet*

Master from Wallet Important Format (WIF).

Parameters *wif* (*str*) – Wallet important format.

Returns *Wallet* – Wallet instance.

```
>>> from pyxdc import HTTP_PROVIDER
>>> from pyxdc.wallet import Wallet
>>> wallet: Wallet = Wallet(provider=HTTP_PROVIDER)
>>> wallet.from_wif(wif="KySR2sF6eTQyYRr3SW12jm5KPycKmgQ9SGUJ7oBQPf1SnvuvJTat
↳")
<pyxdc.wallet.Wallet object at 0x000001E8BFB98D60>
```

from_private_key (*private_key: str*) → *pyxdc.wallet.Wallet*
Master from Private Key.

Parameters **private_key** (*str*) – Private key.

Returns **Wallet** – Wallet instance.

```
>>> from pyxdc import HTTP_PROVIDER
>>> from pyxdc.wallet import Wallet
>>> wallet: Wallet = Wallet(provider=HTTP_PROVIDER)
>>> wallet.from_private_key(private_key=
↳"4235d9ffc246d488d527177b654e7dd5c02f5c5abc2e2054038d6825224a24de")
<pyxdc.wallet.Wallet object at 0x000001E8BFB98D60>
```

from_path (*path: str*) → *pyxdc.wallet.Wallet*
Derivation from Path.

Parameters **path** (*str*) – Derivation path.

Returns **Wallet** – Wallet instance.

```
>>> from pyxdc import HTTP_PROVIDER
>>> from pyxdc.wallet import Wallet
>>> wallet: Wallet = Wallet(provider=HTTP_PROVIDER)
>>> wallet.from_root_xprivate_key(root_xprivate_key=
↳"xprv9s21ZrQH143K3i9qWtFiAawwn2iLAcKKfXHCsTdUsy7RYsAma9qzrrwEwsu9buLocH7qFQmTow5bSysKDmq8V
↳")
>>> wallet.from_path(path="m/44'/550'/'0/0/0")
<pyxdc.wallet.Wallet object at 0x000001E8BFB98D60>
```

from_index (*index: int, hardened: bool = False*) → *pyxdc.wallet.Wallet*
Derivation from Index.

Parameters

- **index** (*int*) – Derivation index.
- **hardened** (*bool*) – Hardened address, default to False.

Returns **Wallet** – Wallet instance.

```
>>> from pyxdc import HTTP_PROVIDER
>>> from pyxdc.wallet import Wallet
>>> wallet: Wallet = Wallet(provider=HTTP_PROVIDER)
>>> wallet.from_root_xprivate_key(root_xprivate_key=
↳"xprv9s21ZrQH143K3i9qWtFiAawwn2iLAcKKfXHCsTdUsy7RYsAma9qzrrwEwsu9buLocH7qFQmTow5bSysKDmq8V
↳")
>>> wallet.from_index(index=44, hardened=True)
>>> wallet.from_index(index=550, hardened=True)
>>> wallet.from_index(index=0, hardened=True)
>>> wallet.from_index(index=0)
>>> wallet.from_index(index=0)
<pyxdc.wallet.Wallet object at 0x000001E8BFB98D60>
```

root_xprivate_key (*encoded: bool = True*) → Optional[str]

Get Root XPrivate Key.

Parameters **encoded** (*bool*) – Encoded root xprivate key, default to True.

Returns str – Root XPrivate Key.

```
>>> from pyxdc import HTTP_PROVIDER
>>> from pyxdc.wallet import Wallet
>>> wallet: Wallet = Wallet(provider=HTTP_PROVIDER)
>>> wallet.from_entropy(entropy="b64dc1c3c3d5b876a94006d49c1e4ed2f106b86e")
>>> wallet.from_path(path="m/44'/550'/0'/0'/0")
>>> wallet.root_xprivate_key()
```

```
↪ "xprv9s21ZrQH143K3i9qWtFiAawwn2iLAcKKfXHCsTdUsy7RYSama9qzrrwEwsu9buLocH7qFQmTow5bSysKDmq8V
↪ "
```

root_xpublic_key (*encoded: bool = True*) → Optional[str]

Get Root XPublic Key.

Parameters **encoded** (*bool*) – Encoded root xpublic key, default to True.

Returns str – Root XPublic Key.

```
>>> from pyxdc import HTTP_PROVIDER
>>> from pyxdc.wallet import Wallet
>>> wallet: Wallet = Wallet(provider=HTTP_PROVIDER)
>>> wallet.from_entropy(entropy="b64dc1c3c3d5b876a94006d49c1e4ed2f106b86e")
>>> wallet.from_path(path="m/44'/550'/0'/0'/0")
>>> wallet.root_xpublic_key()
```

```
↪ "xpub661MyMwAqRbcGCEJcvCiXitgL4Ypa53B2kCofr36SJeQRfVv7hAFQfFio7Qn9R25GrPZZKvvjERGLPBDWxhy
↪ "
```

xprivate_key (*encoded=True*) → Optional[str]

Get XPrivate Key.

Parameters **encoded** (*bool*) – Encoded xprivate key, default to True.

Returns str – Root XPrivate Key.

```
>>> from pyxdc import HTTP_PROVIDER
>>> from pyxdc.wallet import Wallet
>>> wallet: Wallet = Wallet(provider=HTTP_PROVIDER)
>>> wallet.from_entropy(entropy="b64dc1c3c3d5b876a94006d49c1e4ed2f106b86e")
>>> wallet.from_path(path="m/44'/550'/0'/0'/0")
>>> wallet.xprivate_key()
```

```
↪ "xprvA2oDuneWodkNiecDi8VoBCvu7TSfnDmGqr5oKzkkLWvmE9dm1TQzYcp9HZQLqYTep1T3yykxZgiUSJDZYrvnr
↪ "
```

xpublic_key (*encoded: bool = True*) → Optional[str]

Get XPublic Key.

Parameters **encoded** (*bool*) – Encoded xpublic key, default to True.

Returns str – XPublic Key.

```
>>> from pyxdc import HTTP_PROVIDER
>>> from pyxdc.wallet import Wallet
```

(continues on next page)

(continued from previous page)

```
>>> wallet: Wallet = Wallet(provider=HTTP_PROVIDER)
>>> wallet.from_entropy(entropy="b64dc1c3c3d5b876a94006d49c1e4ed2f106b86e")
>>> wallet.from_path(path="m/44'/550'/0'/0/0")
>>> wallet.xpublic_key()

↪ "xpub6FnaKJBQe1Jfw8ggpA2oYLsdfVHABgV8D51Q8PAMtrTk6wxuYzjF6R8d8sX2mAkeqHnGLSugcGDtsLFtmk8pS
↪ "
```

clean_derivation() → *pyxdc.wallet.Wallet*

Clean derivation Path or Indexes.

Returns Wallet – Wallet instance.

```
>>> from pyxdc import HTTP_PROVIDER
>>> from pyxdc.wallet import Wallet
>>> wallet: Wallet = Wallet(provider=HTTP_PROVIDER)
>>> wallet.from_root_xprivate_key(root_xprivate_key=
↪ "xprv9s21ZrQH143K3i9qWtFiAawwn2iLAcKKfXHCsTdUsy7RysAma9qzrrwEwsu9buLocH7qFQmTow5bSysKDmq8V
↪ ")
>>> wallet.from_path(path="m/44'/550'/'0'/0/0")
>>> wallet.path()
"m/44'/550'/'0'/0/0"
>>> wallet.clean_derivation()
<pyxdc.wallet.Wallet object at 0x000001E8BFB98D60>
>>> wallet.path()
None
```

uncompressed() → str

Get Uncompressed Public Key.

Returns str – Uncompressed public key.

```
>>> from pyxdc import HTTP_PROVIDER
>>> from pyxdc.wallet import Wallet
>>> wallet: Wallet = Wallet(provider=HTTP_PROVIDER)
>>> wallet.from_entropy(entropy="b64dc1c3c3d5b876a94006d49c1e4ed2f106b86e")
>>> wallet.from_path(path="m/44'/550'/0'/0/0")
>>> wallet.uncompressed()

↪ "d8799336beacc6b2e7f86f46bce4ad5cabf1ec7a0d6241416985e3b29fe1cc850af47d43f0d7e156dca7e9ab8
↪ "
```

compressed() → str

Get Compressed Public Key.

Returns str – Compressed public key.

```
>>> from pyxdc import HTTP_PROVIDER
>>> from pyxdc.wallet import Wallet
>>> wallet: Wallet = Wallet(provider=HTTP_PROVIDER)
>>> wallet.from_entropy(entropy="b64dc1c3c3d5b876a94006d49c1e4ed2f106b86e")
>>> wallet.from_path(path="m/44'/550'/0'/0/0")
>>> wallet.compressed()
"03d8799336beacc6b2e7f86f46bce4ad5cabf1ec7a0d6241416985e3b29fe1cc85"
```

private_key() → str

Get Private Key.

Returns str – Private key.

```
>>> from pyxdc import HTTP_PROVIDER
>>> from pyxdc.wallet import Wallet
>>> wallet: Wallet = Wallet(provider=HTTP_PROVIDER)
>>> wallet.from_entropy(entropy="b64dc1c3c3d5b876a94006d49c1e4ed2f106b86e")
>>> wallet.from_path(path="m/44'/550'/0'/0/0")
>>> wallet.private_key()
"4235d9ffc246d488d527177b654e7dd5c02f5c5abc2e2054038d6825224a24de"
```

public_key (private_key: Optional[str] = None) → str
Get Public Key.

Returns str – Public key.

```
>>> from pyxdc import HTTP_PROVIDER
>>> from pyxdc.wallet import Wallet
>>> wallet: Wallet = Wallet(provider=HTTP_PROVIDER)
>>> wallet.from_entropy(entropy="b64dc1c3c3d5b876a94006d49c1e4ed2f106b86e")
>>> wallet.from_path(path="m/44'/550'/0'/0/0")
>>> wallet.public_key()
"03d8799336beacc6b2e7f86f46bce4ad5cabf1ec7a0d6241416985e3b29fe1cc85"
```

strength () → Optional[int]
Get Entropy strength.

Returns int – Entropy strength.

```
>>> from pyxdc import HTTP_PROVIDER
>>> from pyxdc.wallet import Wallet
>>> wallet: Wallet = Wallet(provider=HTTP_PROVIDER)
>>> wallet.from_entropy(entropy="b64dc1c3c3d5b876a94006d49c1e4ed2f106b86e")
>>> wallet.strength()
160
```

entropy () → Optional[str]
Get Entropy hex string.

Returns str – Entropy hex string.

```
>>> from pyxdc import HTTP_PROVIDER
>>> from pyxdc.wallet import Wallet
>>> wallet: Wallet = Wallet(provider=HTTP_PROVIDER)
>>> wallet.from_entropy(entropy="b64dc1c3c3d5b876a94006d49c1e4ed2f106b86e")
>>> wallet.entropy()
"b64dc1c3c3d5b876a94006d49c1e4ed2f106b86e"
```

mnemonic () → Optional[str]
Get Mnemonic words.

Returns str – Mnemonic words.

```
>>> from pyxdc import HTTP_PROVIDER
>>> from pyxdc.wallet import Wallet
>>> wallet: Wallet = Wallet(provider=HTTP_PROVIDER)
>>> wallet.from_entropy(entropy="b64dc1c3c3d5b876a94006d49c1e4ed2f106b86e")
>>> wallet.mnemonic()
"venture fitness paper little blush april rigid where find volcano fetch_
↪crack label polar dash"
```

passphrase() → Optional[str]
Get Entropy/Mnemonic passphrase.

Returns str – Entropy/Mnemonic passphrase.

```
>>> from pyxdc import HTTP_PROVIDER
>>> from pyxdc.wallet import Wallet
>>> wallet: Wallet = Wallet(provider=HTTP_PROVIDER)
>>> wallet.from_entropy(entropy="b64dc1c3c3d5b876a94006d49c1e4ed2f106b86e",
↳ passphrase="meherett")
>>> wallet.passphrase()
"meherett"
```

language() → Optional[str]
Get Mnemonic language.

Returns str – Mnemonic language.

```
>>> from pyxdc import HTTP_PROVIDER
>>> from pyxdc.wallet import Wallet
>>> wallet: Wallet = Wallet(provider=HTTP_PROVIDER)
>>> wallet.from_entropy(entropy="b64dc1c3c3d5b876a94006d49c1e4ed2f106b86e")
>>> wallet.language()
"english"
```

seed() → Optional[str]
Get Seed hex string.

Returns str – Seed hex string.

```
>>> from pyxdc import HTTP_PROVIDER
>>> from pyxdc.wallet import Wallet
>>> wallet: Wallet = Wallet(provider=HTTP_PROVIDER)
>>> wallet.from_entropy(entropy="b64dc1c3c3d5b876a94006d49c1e4ed2f106b86e")
>>> wallet.from_path(path="m/44'/550'/0'/0'/0")
>>> wallet.seed()

↳ "09d6f96646d69b3842eecb8f05737972c6c0314d60c203657ae2dad5e8dd88797019ad9938292307de2f4a740"
↳ "
```

path() → Optional[str]
Get Derivation path.

Returns str – Drivation path.

```
>>> from pyxdc import HTTP_PROVIDER
>>> from pyxdc.wallet import Wallet
>>> wallet: Wallet = Wallet(provider=HTTP_PROVIDER)
>>> wallet.from_entropy(entropy="b64dc1c3c3d5b876a94006d49c1e4ed2f106b86e")
>>> wallet.from_path(path="m/44'/550'/0'/0'/0")
>>> wallet.path()
"m/44'/550'/0'/0'/0"
```

chain_code() → Optional[str]
Get Chain code.

Returns str – Chain code.

```
>>> from pyxdc import HTTP_PROVIDER
>>> from pyxdc.wallet import Wallet
>>> wallet: Wallet = Wallet(provider=HTTP_PROVIDER)
>>> wallet.from_entropy(entropy="b64dc1c3c3d5b876a94006d49c1e4ed2f106b86e")
>>> wallet.from_path(path="m/44'/550'/0'/0/0")
>>> wallet.chain_code()
"fb40b46da06b4940be76a38e1962aa34f362c47ccb16707b5e21e71514a98d93"
```

static semantic() → Optional[str]
Get Extended semantic.

Returns str – Extended semantic.

```
>>> from pyxdc import HTTP_PROVIDER
>>> from pyxdc.wallet import Wallet
>>> wallet: Wallet = Wallet(provider=HTTP_PROVIDER)
>>> wallet.from_entropy(entropy="b64dc1c3c3d5b876a94006d49c1e4ed2f106b86e")
>>> wallet.from_path(path="m/44'/550'/0'/0/0")
>>> wallet.semantic()
"p2pkh"
```

hash (*private_key: Optional[str] = None*)
Get Public Key Hash.

Returns str – Identifier/Hash.

```
>>> from pyxdc import HTTP_PROVIDER
>>> from pyxdc.wallet import Wallet
>>> wallet: Wallet = Wallet(provider=HTTP_PROVIDER)
>>> wallet.from_entropy(entropy="b64dc1c3c3d5b876a94006d49c1e4ed2f106b86e")
>>> wallet.from_path(path="m/44'/550'/0'/0/0")
>>> wallet.hash()
"197a8b4ad8fbbel18487e065cc8595bf67845aeb"
```

finger_print() → str
Get Finger print.

Returns str – Finger print.

```
>>> from pyxdc import HTTP_PROVIDER
>>> from pyxdc.wallet import Wallet
>>> wallet: Wallet = Wallet(provider=HTTP_PROVIDER)
>>> wallet.from_entropy(entropy="b64dc1c3c3d5b876a94006d49c1e4ed2f106b86e")
>>> wallet.from_path(path="m/44'/550'/0'/0/0")
>>> wallet.finger_print()
"197a8b4a"
```

address (*prefix: str = 'xdc'*) → str
Get Address.

Returns str – XinFin Address.

```
>>> from pyxdc import HTTP_PROVIDER
>>> from pyxdc.wallet import Wallet
>>> wallet: Wallet = Wallet(provider=HTTP_PROVIDER)
>>> wallet.from_entropy(entropy="b64dc1c3c3d5b876a94006d49c1e4ed2f106b86e")
>>> wallet.from_path(path="m/44'/550'/0'/0/0")
>>> wallet.address()
"xdc9Cd6fD3519b259B251d881361CAae6BABdC5910b"
```


wif() → str

Get Wallet Important Format.

Returns str – Wallet Important Format.

```
>>> from pyxdc import HTTP_PROVIDER
>>> from pyxdc.wallet import Wallet
>>> wallet: Wallet = Wallet(provider=HTTP_PROVIDER)
>>> wallet.from_entropy(entropy="b64dc1c3c3d5b876a94006d49c1e4ed2f106b86e")
>>> wallet.from_path(path="m/44'/550'/0'/0/0")
>>> wallet.wif()
"KySR2sF6eTQyYRr3SW12jm5KPycKmgQ9SGUJ7oBQPf1SnvuvJTat"
```

balance (unit: str = 'Wei') → Union[int, float]

Get XinFin wallet balance.

Parameters unit (str) – XinFin unit, default to Wei.

Returns int, float – XinFin balance (XDC, Gwei, Wei).

```
>>> from pyxdc import HTTP_PROVIDER
>>> from pyxdc.wallet import Wallet
>>> wallet: Wallet = Wallet(provider=HTTP_PROVIDER)
>>> wallet.from_entropy(entropy="b64dc1c3c3d5b876a94006d49c1e4ed2f106b86e")
>>> wallet.from_path(path="m/44'/550'/0'/0/0")
>>> wallet.balance()
2450000000
```

sign (message: Optional[str] = None, message_hash: Optional[str] = None) → str

Sign message data by private key.

Parameters

- **message** (str.) – Message data, default to None.
- **message_hash** (str.) – Message data hash, default to None.

Returns str – Signed message data (signature).

```
>>> from pyxdc import HTTP_PROVIDER
>>> from pyxdc.wallet import Wallet
>>> message =
↳ "1246b84985e1ab5f83f4ec2bdf271114666fd3d9e24d12981a3c861b9ed523c6"
>>> wallet: Wallet = Wallet(provider=HTTP_PROVIDER)
>>> wallet.from_entropy(entropy="b64dc1c3c3d5b876a94006d49c1e4ed2f106b86e")
>>> wallet.from_path(path="m/44'/550'/0'/0/0")
>>> wallet.sign(message=message)
↳ "9c3a1322cab0e70147c85e47bdc3ce7d719130b70857bb7ac633e9bd7a76f3b8d76eddd83f1a5d229a34491b7"
↳ "
```

verify (signature: str, message: Optional[str] = None, message_hash: Optional[str] = None) → bool

Verify signature by public key.

Parameters

- **signature** (str.) – Signed message data.
- **message** (str.) – Message data, default to None.
- **message_hash** (str.) – Message data hash, default to None.

Returns bool – Verified signature (True/False).

```

>>> from pyxdc import HTTP_PROVIDER
>>> from pyxdc.wallet import Wallet
>>> message =
↳ "1246b84985e1ab5f83f4ec2bdf271114666fd3d9e24d12981a3c861b9ed523c6"
>>> signature =
↳ "9c3a1322cab0e70147c85e47bdc3ce7d719130b70857bb7ac633e9bd7a76f3b8d76eddd83f1a5d229a34491b7"
↳ ""
>>> wallet: Wallet = Wallet(provider=HTTP_PROVIDER)
>>> wallet.from_entropy(entropy="b64dc1c3c3d5b876a94006d49c1e4ed2f106b86e")
>>> wallet.from_path(path="m/44'/550'/0'/0/0")
>>> wallet.verify(message=message, signature=signature)
True

```

dumps() → dict

Get All Wallet informations.

Returns dict – All Wallet informations.

```

>>> from pyxdc import HTTP_PROVIDER
>>> from pyxdc.wallet import Wallet
>>> wallet: Wallet = Wallet(provider=HTTP_PROVIDER)
>>> wallet.from_entropy(entropy="b64dc1c3c3d5b876a94006d49c1e4ed2f106b86e")
>>> wallet.from_path(path="m/44'/550'/0'/0/0")
>>> wallet.dumps()
{'strength': 160, 'entropy': 'b64dc1c3c3d5b876a94006d49c1e4ed2f106b86e',
↳ 'mnemonic': 'rent host ill marble fortune deputy pink absorb stand thought',
↳ 'neck planet away found robust', 'language': 'english', 'passphrase': None,
↳ 'seed':
↳ '09d6f96646d69b3842eecb8f05737972c6c0314d60c203657ae2dad5e8dd88797019ad9938292307de2f4a740',
↳ ', 'root_xprivate_key':
↳ 'xprv9s21ZrQH143K3i9qWtFiAawwn2iLAcKKfXHCsTdUsy7RYsAma9qzrrwEwsu9buLocH7qFQmTow5bSysKDmq8V',
↳ ', 'root_xpublic_key':
↳ 'xpub661MyMwAqRbcGCEJcvCiXitgL4Ypa53B2kCoFr36SJeQRfVv7hAFQfFio7Qn9R25GrPZZKvvjERGLPBTDWxhy',
↳ ', 'xprivate_key':
↳ 'xprvA2oDuneWodkNieCDi8VoBCvu7TSfnDmGqr5oKzkkLWvME9dm1TQzYcp9HZQLqYTep1T3yykxZgiUSJDZYrvnr',
↳ ', 'xpublic_key':
↳ 'xpub6FnaKJBQe1Jfw8ggpA2oYLsdfVHABgV8D51Q8PAMtrTk6wxuYzjF6R8d8sX2mAkeqHnGLSugcGDtsLFtmk8pS',
↳ ', 'uncompressed':
↳ 'd8799336beacc6b2e7f86f46bce4ad5cabf1ec7a0d6241416985e3b29fe1cc850af47d43f0d7e156dca7e9ab8',
↳ ', 'compressed':
↳ '03d8799336beacc6b2e7f86f46bce4ad5cabf1ec7a0d6241416985e3b29fe1cc85',
↳ 'chain_code':
↳ 'fb40b46da06b4940be76a38e1962aa34f362c47ccb16707b5e21e71514a98d93',
↳ 'private_key':
↳ '4235d9ffc246d488d527177b654e7dd5c02f5c5abc2e2054038d6825224a24de', 'public_
↳ 'key': '03d8799336beacc6b2e7f86f46bce4ad5cabf1ec7a0d6241416985e3b29fe1cc85',
↳ 'wif': 'KySR2sF6eTQyYRr3SW12jm5KPycKmgQ9SGUJ7oBQPf1SnvuvJTat', 'finger_print
↳ ': '197a8b4a', 'semantic': 'p2pkh', 'path': "m/44'/550'/0'/0/0", 'hash':
↳ '197a8b4ad8fbbel18487e065cc8595bf67845aeb', 'address':
↳ 'xdc9Cd6fD3519b259B251d881361CAae6BABdC5910b'}

```

RPC

```
pyxdc.rpc.get_balance(address: str, provider: Union[web3.providers.rpc.HTTPProvider,
web3.providers.websocket.WebsocketProvider] =
<web3.providers.rpc.HTTPProvider object>, unit: str = 'Wei') → Union[int,
float]
```

Get XinFin balance.

Parameters

- **address** (*str*) – XinFin address.
- **provider** (*HTTPProvider*, *WebsocketProvider*) – XinFin provider, default to HTTP_PROVIDER.
- **unit** (*str*) – XinFin unit, default to Wei.

Returns int, float – XinFin balance (XDC, Gwei, Wei).

```
>>> from pyxdc import HTTP_PROVIDER
>>> from pyxdc.rpc import get_balance
>>> get_balance(address="xdc70c1eb09363603a3b6391deb2daa6d2561a62f52",
↪ provider=HTTP_PROVIDER)
71560900
```

```
pyxdc.rpc.get_transaction(transaction_hash: str, headers: dict = {'Accept': 'application/json',
'Content-Type': 'application/json; charset=utf-8', 'User-Agent': 'PyXDC
User-Agent v0.1.0'}, timeout: int = 60) → dict
```

Get XinFin transaction detail.

Parameters

- **transaction_hash** (*str*) – XinFin transaction hash.
- **headers** (*dict*) – Request headers, default to common headers.
- **timeout** (*int*) – Request timeout, default to 15.

Returns dict – XinFin transaction detail.

```
>>> from pyxdc.rpc import get_transaction
>>> get_transaction(transaction_hash=
↪ "0xa4d57071427e3310b3e2fb16e7712f8d8aaaafb31ce5fcd6534fc50848905948")
{'hash': '0xa4d57071427e3310b3e2fb16e7712f8d8aaaafb31ce5fcd6534fc50848905948',
↪ 'nonce': 0, 'blockHash':
↪ '0xb33a804ae10713bf549db8ec749f7d650347613ac784db1a8d17e0cb03741bf0',
↪ 'blockNumber': 1, 'transactionIndex': 0, 'from':
↪ '0x96cA14396341480E3b6384D1d1397d1f7f5a0AB7', 'to': None, 'value': 0, 'gas':
↪ 367400, 'gasPrice': 250000000, 'input':
↪ '0x608060405234801561001057600080fd5b506040518060400160405280600581526020017f48656c6c6f000000
(continues on next page)
↪ ', 'v': 28, 'r':
↪ '0xa593dcfd7f7b17f8b22907e9c4b03721312a4d00dfd99f8f7267ccd5eb7d4613', 's':
↪ '0x70cd172ae92de7a046dfe28de1db8657f8c3b3ed00c060392fb1d5080646927b'}
```

(continued from previous page)

```
pyxdc.rpc.submit_transaction_raw(transaction_raw: str, provider:
                                Union[web3.providers.rpc.HTTPProvider,
                                web3.providers.websocket.WebsocketProvider] =
                                <web3.providers.rpc.HTTPProvider object>) → str
```

Submit XinFin transaction raw.

Parameters

- **transaction_raw** (*str*) – XinFin transaction raw.
- **provider** (*HTTPProvider*, *WebsocketProvider*) – XinFin provider, default to HTTP_PROVIDER.

Returns *str* – XinFin submitted transaction hash.

```
>>> from pyxdc import HTTP_PROVIDER
>>> from pyxdc.rpc import submit_transaction_raw
>>> submit_transaction_raw(transaction_raw=
↳ "0xf86c02840ee6b280825208943e0a9b2ee8f8341a1aead3e7531d75f1e395f24b8901236efcbcb340000801ba03
↳ ", provider=HTTP_PROVIDER)
"0x04b3bfb804f2b3329555c6f3a17a794b3f099b6435a9cf58c78609ed93853907"
```

TRANSACTION

```
class pyxdc.transaction.Transaction(provider: Union[web3.providers.rpc.HTTPProvider,  
web3.providers.websocket.WebsocketProvider] =  
<web3.providers.rpc.HTTPProvider object>)
```

XinFin Transaction.

Parameters **provider** (*HTTPProvider*, *WebsocketProvider*) – XinFin provider, default to HTTP_PROVIDER.

Returns Transaction – XinFin transaction instance.

fee (*unit: str = 'Wei'*) → Union[int, float]
Get XinFin transaction fee/gas.

Parameters **unit** (*str*) – XinFin unit, default to Wei.

Returns int, float – XinFin transaction fee/gas.

```
>>> from pyxdc import HTTP_PROVIDER
>>> from pyxdc.transaction import NormalTransaction
>>> transaction: NormalTransaction = NormalTransaction(provider=HTTP_PROVIDER)
>>> transaction.build_transaction(...)
>>> transaction.fee()
367400
```

hash () → str
Get XinFin transaction hash.

Returns str – XinFin transaction hash.

```
>>> from pyxdc import HTTP_PROVIDER
>>> from pyxdc.transaction import NormalTransaction
>>> transaction: NormalTransaction = NormalTransaction(provider=HTTP_PROVIDER)
>>> transaction.build_transaction(...)
>>> transaction.hash()
"2993414225f65390220730d0c1a356c14e91bca76db112d37366df93e364a492"
```

json () → dict
Get XinFin transaction json format.

Returns dict – XinFin transaction json format.

```
>>> from pyxdc import WEBSOCKET_PROVIDER
>>> from pyxdc.transaction import ContractTransaction
>>> transaction: ContractTransaction = ContractTransaction(provider=WEBSOCKET_
↳ PROVIDER)
>>> transaction.build_transaction(...)
```

(continues on next page)

(continued from previous page)

[illegible]

raw() → str

Get XinFin transaction raw.

Returns str – XinFin transaction raw.

```
>>> from pyxdc import HTTP_PROVIDER
>>> from pyxdc.transaction import ContractTransaction
>>> transaction: ContractTransaction = ContractTransaction(provider=HTTP_
↳ PROVIDER)
>>> transaction.build_transaction(...)
>>> transaction.raw()

↳ "f8cc0f8504a817c80083020cb0949f77b9f27e8bc8ad0b58fbf99aea28feec7ec50b64b864335ef5bd0000000
↳ "
```

5.1 ContractTransaction

```
class pyxdc.transaction.ContractTransaction (provider: Union[web3.providers.rpc.HTTPProvider,  
web3.providers.websocket.WebsocketProvider]  
= <web3.providers.rpc.HTTPProvider ob-  
ject>)
```

XinFin Contract Transaction.

Parameters `provider` (*HTTPProvider*, *WebsocketProvider*) – XinFin provider, default to `HTTP_PROVIDER`.

Returns ContractTransaction – XinFin contract transaction instance.

```
build_transaction(address: str, abi: list, bytecode: str, value: int = 0, gas: Optional[int] = None,
    estimate_gas: bool = True, gas_price: int = 250000000, *args, **kwargs) →
    pyxdc.transaction.ContractTransaction
```

Build XinFin contract transaction.

Parameters

- **address** (*str*) – XinFin from address.
- **abi** (*list*) – XinFin smart contract abi.
- **bytecode** (*str*) – XinFin smart contract bytecode.
- **value** (*int*) – XinFin Wei value, defaults to 0.
- **gas** (*int*) – XinFin transaction fee/gas, defaults to None.
- **estimate_gas** (*bool*) – XinFin transaction estimate fee/gas, defaults to True.
- **gas_price** (*int*) – XinFin gas price, defaults to 0.25 Gwei.

Returns ContractTransaction – XinFin contract transaction instance.

```
>>> from pyxdc import WEBSOCKET_PROVIDER
>>> from pyxdc.transaction import ContractTransaction
>>> contract_transaction: ContractTransaction =
↳ ContractTransaction(provider=WEBSOCKET_PROVIDER)
>>> contract_transaction.build_transaction(address=
↳ "xdc571ae1504e92fa40f85359efdb188c704a224eac", abi=[...], bytecode="...",
↳ value=0, estimate_gas=True)
<pyxdc.transaction.ContractTransaction object at 0x0409DAF0>
```

sign_transaction (*private_key: Optional[str] = None, root_xprivate_key: Optional[str] = None, path: str = "m/44'/550'/0'/0/0"*) → *pyxdc.transaction.ContractTransaction*
Sign XinFin contract transaction.

Parameters

- **private_key** (*str*) – XinFin private key, default to None.
- **root_xprivate_key** (*str*) – XinFin root xprivate key, default to None.
- **path** (*str*) – XinFin derivation path, default to DEFAULT_PATH.

Returns *ContractTransaction* – Signed XinFin contract transaction instance.

```
>>> from pyxdc import WEBSOCKET_PROVIDER
>>> from pyxdc.transaction import ContractTransaction
>>> contract_transaction: ContractTransaction =
↳ ContractTransaction(provider=WEBSOCKET_PROVIDER)
>>> contract_transaction.build_transaction(address=
↳ "xdc571ae1504e92fa40f85359efdb188c704a224eac", abi=[...], bytecode="...",
↳ value=0, estimate_gas=True)
>>> contract_transaction.sign_transaction(private_key=
↳ "4235d9ffc246d488d527177b654e7dd5c02f5c5abc2e2054038d6825224a24de")
<pyxdc.transaction.ContractTransaction object at 0x0409DAF0>
```

5.2 NormalTransaction

class *pyxdc.transaction.NormalTransaction* (*provider: Union[web3.providers.rpc.HTTPProvider, web3.providers.websocket.WebsocketProvider] = <web3.providers.rpc.HTTPProvider object>*)

XinFin Normal Transaction.

Parameters **provider** (*HTTPProvider, WebsocketProvider*) – XinFin provider, default to HTTP_PROVIDER.

Returns *NormalTransaction* – XinFin normal transaction instance.

build_transaction (*address: str, recipient: str, value: int, gas: Optional[int] = None, estimate_gas: bool = True, gas_price: int = 250000000*) → *pyxdc.transaction.NormalTransaction*
Build XinFin normal transaction.

Parameters

- **address** (*str*) – XinFin from address.
- **recipient** (*str*) – Recipients XinFin address.
- **value** (*int*) – XinFin Wei value.

- **gas** (*int*) – XinFin transaction fee/gas, defaults to None.
- **estimate_gas** (*bool*) – XinFin transaction estimate fee/gas, defaults to True.
- **gas_price** (*int*) – XinFin gas price, defaults to 0.25 Gwei.

Returns NormalTransaction – XinFin normal transaction instance.

```
>>> from pyxdc import HTTP_PROVIDER
>>> from pyxdc.transaction import NormalTransaction
>>> normal_transaction: NormalTransaction = NormalTransaction(provider=HTTP_
↳ PROVIDER)
>>> normal_transaction.build_transaction(address=
↳ "xdc571ae1504e92fa40f85359efdb188c704a224eac", recipient=
↳ "xdc3e0a9B2Ee8F8341A1aEaD3E7531d75f1e395F24b", value=1_000_000_000,
↳ estimate_gas=True)
<pyxdc.transaction.NormalTransaction object at 0x0409DAF0>
```

sign_transaction (*private_key: Optional[str] = None, root_xprivate_key: Optional[str] = None, path: str = "m/44'/550'/0'/0/'0") → pyxdc.transaction.NormalTransaction*
Sign XinFin normal transaction.

Parameters

- **private_key** (*str*) – XinFin private key, default to None.
- **root_xprivate_key** (*str*) – XinFin root xprivate key, default to None.
- **path** (*str*) – XinFin derivation path, default to DEFAULT_PATH.

Returns NormalTransaction – Signed XinFin normal transaction instance.

```
>>> from pyxdc import HTTP_PROVIDER, DEFAULT_PATH
>>> from pyxdc.transaction import NormalTransaction
>>> normal_transaction: NormalTransaction = NormalTransaction(provider=HTTP_
↳ PROVIDER)
>>> normal_transaction.build_transaction(address=
↳ "xdc571ae1504e92fa40f85359efdb188c704a224eac", recipient=
↳ "xdc3e0a9B2Ee8F8341A1aEaD3E7531d75f1e395F24b", value=1_000_000_000,
↳ estimate_gas=True)
>>> normal_transaction.sign_transaction(root_xprivate_key=
↳ "xprv9s21ZrQH143K3i9qWtfiAawwn2iLAcKKfXHCsTdUsy7RYsAma9qzrrwEwsu9buLocH7qFQmTow5bSysKDmq8V
↳ ", path=DEFAULT_PATH)
<pyxdc.transaction.NormalTransaction object at 0x0409DAF0>
```


SIGNATURE

`pyxdc.signature.sign` (*private_key: str, message: Optional[str] = None, message_hash: Optional[str] = None*) → *str*
 Sign XinFin message data by private key.

Parameters

- **private_key** (*str.*) – XinFin private key.
- **message** (*str.*) – Message data, default to None.
- **message_hash** (*str.*) – Message data hash, default to None.

Returns *str* – XinFin signed message or signature.

```
>>> from pyxdc.signature import sign
>>> sign(private_key=
↳ "4235d9ffc246d488d527177b654e7dd5c02f5c5abc2e2054038d6825224a24de", message=
↳ "meherett")

↳ "74ad07a84b87fa3efa2f0e825506fb8bbe41021ca77a30e8ffa2bd66d47d99917d4a0587185e78a051a9cb80ebf6
↳ "
>>> sign(private_key=
↳ "4235d9ffc246d488d527177b654e7dd5c02f5c5abc2e2054038d6825224a24de", message_
↳ hash="4bbbfd0c33fea618f4a9aa75c02fe76e50fa59798af021bc34f7856f3259c685")

↳ "74ad07a84b87fa3efa2f0e825506fb8bbe41021ca77a30e8ffa2bd66d47d99917d4a0587185e78a051a9cb80ebf6
↳ "
```

`pyxdc.signature.verify` (*public_key: str, signature: str, message: Optional[str] = None, message_hash: Optional[str] = None*) → *bool*
 Verify XinFin signature by public key.

Parameters

- **public_key** (*str.*) – XinFin public key.
- **signature** (*str.*) – Signed message data.
- **message** (*str.*) – Message data, default to None.
- **message_hash** (*str.*) – Message data hash, default to None.

Returns *bool* – Verified signature.

```
>>> from pyxdc.signature import verify
>>> verify(public_key=
↳ "03d8799336beacc6b2e7f86f46bce4ad5cabf1ec7a0d6241416985e3b29fe1cc85", message=
↳ "meherett", signature=
↳ "74ad07a84b87fa3efa2f0e825506fb8bbe41021ca77a30e8ffa2bd66d47d99917d4a0587185e78a051a9cb80ebf6
↳ ")
(continues on next page)
```

(continued from previous page)

```
True
>>> verify(public_key=
↳ "03d8799336beacc6b2e7f86f46bce4ad5cabf1ec7a0d6241416985e3b29fe1cc85", message_
↳ hash="4bbbfd0c33fea618f4a9aa75c02fe76e50fa59798af021bc34f7856f3259c685",
↳ signature=
↳ "74ad07a84b87fa3efa2f0e825506fb8bbe41021ca77a30e8ffa2bd66d47d99917d4a0587185e78a051a9cb80ebf6
↳ ")
True
```

UTILS

`pyxdc.utils.generate_passphrase (length: int = 32) → str`
Generate entropy hex string.

Parameters `length (int)` – Passphrase length, default to 32.

Returns `str` – Passphrase hex string.

```
>>> from pyxdc.utils import generate_passphrase
>>> generate_passphrase(length=32)
"N39rPf3QvF2Tm2nPyoBpXNiBFXJywTz"
```

`pyxdc.utils.generate_entropy (strength: int = 128) → str`
Generate entropy hex string.

Parameters `strength (int)` – Entropy strength, default to 128.

Returns `str` – Entropy hex string.

```
>>> from pyxdc.utils import generate_entropy
>>> generate_entropy(strength=128)
"ee535b143b0d9d1f87546f9df0d06b1a"
```

`pyxdc.utils.generate_mnemonic (language: str = 'english', strength: int = 128) → str`
Generate mnemonic words.

Parameters

- **language (str)** – Mnemonic language, default to english.
- **strength (int)** – Entropy strength, default to 128.

Returns `str` – Mnemonic words.

```
>>> from pyxdc.utils import generate_mnemonic
>>> generate_mnemonic(language="french")
"sceptre capter sequence girafe absolu relatif fleur zoologie muscle sirop_
↳saboter parure"
```

`pyxdc.utils.is_entropy (entropy: str) → bool`
Check entropy hex string.

Parameters `entropy (str)` – Mnemonic words.

Returns `bool` – Entropy valid/invalid.

```
>>> from pyxdc.utils import is_entropy
>>> is_entropy(entropy="ee535b143b0d9d1f87546f9df0d06b1a")
True
```

`pyxdc.utils.is_mnemonic(mnemonic: str, language: Optional[str] = None) → bool`

Check mnemonic words.

Parameters

- **mnemonic** (*str*) – Mnemonic words.
- **language** (*str*) – Mnemonic language, default to None.

Returns bool – Mnemonic valid/invalid.

```
>>> from pyxdc.utils import is_mnemonic
>>> is_mnemonic(mnemonic="sceptre capter sequence girafe absolu relatif fleur_
↳ zoologie muscle sirop saboter parure")
True
```

`pyxdc.utils.is_address(address: str) → bool`

Check XinFin address.

Parameters **address** (*str*) – XinFin address.

Returns bool – XinFin valid/invalid address.

```
>>> from pyxdc.utils import is_address
>>> is_address(address="xdc1ee11011ae12103a488a82dc33e03f337bc93ba7")
True
```

`pyxdc.utils.is_checksum_address(address: str) → bool`

Check XinFin checksum address.

Parameters **address** (*str*) – XinFin address.

Returns bool – XinFin valid/invalid checksum address.

```
>>> from pyxdc.utils import is_checksum_address
>>> is_checksum_address(address="xdc1ee11011ae12103a488a82dc33e03f337bc93ba7")
False
>>> is_checksum_address(address="xdc1Ee11011ae12103a488A82DC33e03f337Bc93ba7")
True
```

`pyxdc.utils.to_checksum_address(address: str, prefix: str = 'xdc') → str`

To XinFin checksum address.

Parameters

- **address** (*str*) – XinFin address.
- **prefix** (*str*) – XinFin address prefix, default to xdc.

Returns str – XinFin checksum address.

```
>>> from pyxdc.utils import is_checksum_address
>>> is_checksum_address(address="xdc1ee11011ae12103a488a82dc33e03f337bc93ba7")
"xdc1Ee11011ae12103a488A82DC33e03f337Bc93ba7"
```

`pyxdc.utils.decode_transaction_raw(transaction_raw: str) → dict`

Decode XinFin transaction raw.

Parameters **transaction_raw** (*str*) – XinFin transaction raw.

Returns dict – XinFin decoded transaction.

```
>>> from pyxdc.utils import decode_transaction_raw
>>> decode_transaction_raw(transaction_raw=
↳ "0xf90703058504a817c800831e84808080b906b0608060405234801561001057600080fd5b5060405180604001604
↳ ")
{'hash': '0x57232e7e3f0e4f5f49cad5074bea10c98ee18efd4371e15c163560b8bc8ebb40',
↳ 'from': '0x68bF25F60508C2820d3D72E1806503F0955eFf94', 'to': None, 'nonce': 5,
↳ 'gas': 2000000, 'gas_price': 20000000000, 'value': 0, 'data':
↳ '0x608060405234801561001057600080fd5b506040518060400160405280600581526020017f48656c6c6f0000000
↳ ', 'chain_id': -4, 'r':
↳ '0xf2704e20656acf4b067c23ff6e7e2bf8e9b6f75383c408607fce7f90ef39aedb', 's':
↳ '0x7612be142f5202b3970ee9b4c821bd95df4eb007735acc9c145b0d204d697f8c', 'v': 28}
```

`pyxdc.utils.get_entropy_strength(entropy: str) → int`

Get entropy strength.

Parameters `entropy` (*str*) – Entropy hex string.

Returns `int` – Entropy strength.

```
>>> from pyxdc.utils import get_entropy_strength
>>> get_entropy_strength(entropy="ee535b143b0d9d1f87546f9df0d06b1a")
128
```

`pyxdc.utils.get_mnemonic_strength(mnemonic: str, language: Optional[str] = None) → int`

Get mnemonic strength.

Parameters

- **mnemonic** (*str*) – Mnemonic words.
- **language** (*str*) – Mnemonic language, default to None.

Returns `int` – Mnemonic strength.

```
>>> from pyxdc.utils import get_mnemonic_strength
>>> get_mnemonic_strength(mnemonic="sceptre capter sequence girafe absolu relatif_
↳ fleur zoologie muscle sirop saboter parure")
128
```

`pyxdc.utils.get_mnemonic_language(mnemonic: str) → str`

Get mnemonic language.

Parameters `mnemonic` (*str*) – Mnemonic words.

Returns `str` – Mnemonic language.

```
>>> from pyxdc.utils import get_mnemonic_language
>>> get_mnemonic_language(mnemonic="sceptre capter sequence girafe absolu relatif_
↳ fleur zoologie muscle sirop saboter parure")
"french"
```

`pyxdc.utils.entropy_to_mnemonic(entropy: str, language: str = 'english') → str`

Get mnemonic from entropy hex string.

Parameters

- **entropy** (*str*) – Entropy hex string.
- **language** (*str*) – Mnemonic language, default to english.

Returns `str` – Mnemonic words.

```
>>> from pyxdc.utils import entropy_to_mnemonic
>>> entropy_to_mnemonic(entropy="ee535b143b0d9d1f87546f9df0d06b1a", language=
↪ "korean")
"
```

`pyxdc.utils.mnemonic_to_entropy` (*mnemonic: str, language: Optional[str] = None*) → *str*
Get entropy from mnemonic words.

Parameters

- **mnemonic** (*str*) – Mnemonic words.
- **language** (*str*) – Mnemonic language, default to english.

Returns *str* – Entropy hex string.

```
>>> from pyxdc.utils import mnemonic_to_entropy
>>> mnemonic_to_entropy(mnemonic="", language="korean")
"ee535b143b0d9d1f87546f9df0d06b1a"
```

`pyxdc.utils.amount_unit_converter` (*amount: Union[int, float], unit: str = 'Wei2XDC'*) → *Union[int, float]*
XinFin amount unit converter.

Parameters

- **amount** (*int, float*) – XinFin amount.
- **unit** (*str*) – XinFin unit, default to Wei2XDC

Returns *int, float* – XinFin amount.

```
>>> from pyxdc.utils import amount_unit_converter
>>> amount_unit_converter(amount=100_000_000, unit="Wei2XDC")
0.1
```

PYTHON MODULE INDEX

p

`pyxdc.rpc`, [15](#)
`pyxdc.signature`, [21](#)
`pyxdc.utils`, [23](#)

A

`address()` (*pyxdc.wallet.Wallet method*), 12
`amount_unit_converter()` (in module *pyxdc.utils*), 26

B

`balance()` (*pyxdc.wallet.Wallet method*), 13
`build_transaction()`
 (*pyxdc.transaction.ContractTransaction method*), 18
`build_transaction()`
 (*pyxdc.transaction.NormalTransaction method*), 19

C

`chain_code()` (*pyxdc.wallet.Wallet method*), 11
`clean_derivation()` (*pyxdc.wallet.Wallet method*), 9
`compressed()` (*pyxdc.wallet.Wallet method*), 9
`ContractTransaction` (class in *pyxdc.transaction*), 18

D

`decode_transaction_raw()` (in module *pyxdc.utils*), 24
`dumps()` (*pyxdc.wallet.Wallet method*), 14

E

`entropy()` (*pyxdc.wallet.Wallet method*), 10
`entropy_to_mnemonic()` (in module *pyxdc.utils*), 25

F

`fee()` (*pyxdc.transaction.Transaction method*), 17
`finger_print()` (*pyxdc.wallet.Wallet method*), 12
`from_entropy()` (*pyxdc.wallet.Wallet method*), 5
`from_index()` (*pyxdc.wallet.Wallet method*), 7
`from_mnemonic()` (*pyxdc.wallet.Wallet method*), 5
`from_path()` (*pyxdc.wallet.Wallet method*), 7
`from_private_key()` (*pyxdc.wallet.Wallet method*), 7

`from_root_xprivate_key()` (*pyxdc.wallet.Wallet method*), 6
`from_seed()` (*pyxdc.wallet.Wallet method*), 6
`from_wif()` (*pyxdc.wallet.Wallet method*), 6
`from_xprivate_key()` (*pyxdc.wallet.Wallet method*), 6

G

`generate_entropy()` (in module *pyxdc.utils*), 23
`generate_mnemonic()` (in module *pyxdc.utils*), 23
`generate_passphrase()` (in module *pyxdc.utils*), 23
`get_balance()` (in module *pyxdc.rpc*), 15
`get_entropy_strength()` (in module *pyxdc.utils*), 25
`get_mnemonic_language()` (in module *pyxdc.utils*), 25
`get_mnemonic_strength()` (in module *pyxdc.utils*), 25
`get_transaction()` (in module *pyxdc.rpc*), 15

H

`hash()` (*pyxdc.transaction.Transaction method*), 17
`hash()` (*pyxdc.wallet.Wallet method*), 12

I

`is_address()` (in module *pyxdc.utils*), 24
`is_checksum_address()` (in module *pyxdc.utils*), 24
`is_entropy()` (in module *pyxdc.utils*), 23
`is_mnemonic()` (in module *pyxdc.utils*), 23

J

`json()` (*pyxdc.transaction.Transaction method*), 17

L

`language()` (*pyxdc.wallet.Wallet method*), 11

M

`mnemonic()` (*pyxdc.wallet.Wallet method*), 10
`mnemonic_to_entropy()` (in module *pyxdc.utils*), 26

module

 pyxdc.rpc, 15
 pyxdc.signature, 21
 pyxdc.utils, 23

N

NormalTransaction (*class in pyxdc.transaction*), 19

P

passphrase() (*pyxdc.wallet.Wallet method*), 10
path() (*pyxdc.wallet.Wallet method*), 11
private_key() (*pyxdc.wallet.Wallet method*), 9
public_key() (*pyxdc.wallet.Wallet method*), 10
pyxdc.rpc
 module, 15
pyxdc.signature
 module, 21
pyxdc.utils
 module, 23

R

raw() (*pyxdc.transaction.Transaction method*), 18
root_xprivate_key() (*pyxdc.wallet.Wallet method*), 7
root_xpublic_key() (*pyxdc.wallet.Wallet method*), 8

S

seed() (*pyxdc.wallet.Wallet method*), 11
semantic() (*pyxdc.wallet.Wallet static method*), 12
sign() (*in module pyxdc.signature*), 21
sign() (*pyxdc.wallet.Wallet method*), 13
sign_transaction()
 (*pyxdc.transaction.ContractTransaction method*), 19
sign_transaction()
 (*pyxdc.transaction.NormalTransaction method*), 20
strength() (*pyxdc.wallet.Wallet method*), 10
submit_transaction_raw() (*in module pyxdc.rpc*), 16

T

to_checksum_address() (*in module pyxdc.utils*), 24
Transaction (*class in pyxdc.transaction*), 17

U

uncompressed() (*pyxdc.wallet.Wallet method*), 9

V

verify() (*in module pyxdc.signature*), 21
verify() (*pyxdc.wallet.Wallet method*), 13

W

Wallet (*class in pyxdc.wallet*), 5
wif() (*pyxdc.wallet.Wallet method*), 12

X

xprivate_key() (*pyxdc.wallet.Wallet method*), 8
xpublic_key() (*pyxdc.wallet.Wallet method*), 8