wavespy

DPInvaders

Sep 12, 2021

CONTENTS:

`wavespy` is Python API client for Waves node. It provides sync HTTP client based on requests module and async based on aiohttp. Also it has some useful features e.g. Waves address generation and validation, transaction data generation.

CHAPTER

ONE

WAVESPY?

Originally pyacryl2

CHAPTER

TWO

INSTALLATION

```
pip install wavespy
```

From source:

```
python setup.py install
```

Tests:

From source:

```
python setup.py test
```

CHAPTER

THREE

MAKING REQUESTS TO NODE API

Using sync client:

```
from wavespy.client import WavesClient
client = WavesClient()
print(client.node_version())
```

Or using async client:

```
import asyncio
from wavespy.async_client import WavesAsyncClient
client = WavesAsyncClient()

# Python 3.6
loop = asyncio.get_event_loop()
print(loop.run_until_complete(client.node_version()))
# Python 3.7
print(asyncio.run(client.node_version()))
```

3.1 API clients

`wavespy` provides sync client *`WavesClient`* and async *`WavesAsyncClient`*

3.1.1 Sync client

To make a request with sync client:

```
from wavespy.client import WavesClient

client = WavesClient()
node_version = client.node_version()
```

If request is successful you can get response data:

```
print(node_version.response_data)
```

Get response data items or iterate over response data:

```
print(node_version['version'])
```

3.1.2 Async client

To make a request with async client:

```
import asyncio
from wavespy.client import WavesAsyncClient

async_client = WavesAsyncClient()
node_version = asyncio.run(async_client.node_version())
```

3.1.3 Session reuse

By default API requests made in a new session. You can reuse same session with *start_session()* method:

```
import requests
from wavespy import WavesClient

client = WavesClient()
client.start_session()
node_version = client.node_version()
node_time = client.utils_time()
client.close_session()
```

For async:

```
import asyncio
from aiohttp import ClientSession
from wavespy import WavesAsyncClient

loop = asyncio.get_event_loop()
async_client = WavesAsyncClient()
loop.run_until_complete(async_client.start_session())
node_version = loop.run_until_complete(async_client.node_version())
node_time = loop.run_until_complete(async_client.utils_time())
loop.run_until_complete(async_client.close_session())
```

Or using context. For sync client:

```
from wavespy import WavesClient

with WavesClient() as client:
    node_version = client.node_version()
    node_time = client.utils_time()
```

For async client:

```
import asyncio
from wavespy import WavesAsyncClient

async def get_data():
    async with WavesAsyncClient() as async_client:
        node_version = await async_client.node_version()
        node_time = await async_client.utils_time()

    return (node_version, node_time)
```

(continues on next page)

(continued from previous page)

```
loop = asyncio.get_event_loop()
result = loop.run_until_complete(get_data())
```

3.2 Utilities

With *wavespy.utils* you can easily create or verify Waves addresses, generate transaction data and simplify API requests.

3.2.1 Addresses

With *WavesAddress* you can create, sign and broadcast transactions. For example Waves transfer transaction:

```
from wavespy.utils import WavesAddress
address = WavesAddress('address_base58_value', 'address_private_key')
print(address.transfer_waves('recipient_address', 1000))
```

Also it can be used to receive information from node, e.g. balance:

```
from wavespy.utils import WavesAddress
address = WavesAddress('address_base58_value', 'address_private_key')
print(address.get_balance())
```

`AsyncWavesAddress` provides the same functionality as *WavesAddress* but with asynchronous API client:

```
from wavespy.utils import WavesAddress
address = AsyncWavesAddress('address_base58_value', 'address_private_key')
loop = asyncio.get_event_loop()
result = loop.run_until_complete(
    address.transfer_waves('recipient_address', 1000))
print(result)
```

3.2.2 Address generator

`WavesAddressGenerator` provides functionality for address creation and verification. Create new Waves address:

```
from wavespy.utils import WavesAddressGenerator
new_address = WavesAddressGenerator()
```

Create from existing seed, private or public key:

```
address = WavesAddressGenerator().generate(seed="your_seed_here")
address = WavesAddressGenerator().generate(private_key="your_private_key")
address = WavesAddressGenerator().generate(public_key="your_public_key")
```

Validate address and get address object:

```
address_generator = WavesAddressGenerator()
address = address_generator.generate(value="address", private_key="your_private_key",
                                     public_key="your_public_key")
```

By default `generate()` returns *WavesAddress* object. If you need address object with async API client `WavesAsyncAddress`

```
async_generator = WavesAddressGenerator(async_address=True)
new_async_address = async_generator.generate()
```

3.3 wavespy API

3.3.1 *client* module

wavespy.client

This module provides API client class

class wavespy.client.**BaseClient**(*node_address='https://nodes.wavesplatform.com'*, *matcher_address='https://matcher.waves.exchange'*, *chain_id=None*, *api_key=None*, *raise_exception=True*, *request_params=None*, *online=True*)

Bases: `object`

Base class for API clients

Parameters

- **node_address** (*str*) – node url
- **matcher_address** (*str*) – node url
- **chain_id** (*str*) – chain id
- **api_key** (*str*) – API key for private methods
- **raise_exception** (*bool*) – raise WavesClientException on request error
- **request_params** (*dict*) – request params dict e.g. timeout, proxies. May vary by client, see client lib docs
- **online** (*bool*) – send requests to node if true, else return prepared request with data

class wavespy.client.**WavesClient**(*node_address='https://nodes.wavesplatform.com'*, *matcher_address='https://matcher.waves.exchange'*, *chain_id=None*, *api_key=None*, *raise_exception=True*, *request_params=None*, *online=True*)

Bases: *wavespy.client.BaseClient*

Waves API client class based on *requests* HTTP client Class method names usually consist of a API endpoint, like */blocks/height* is *blocks_height* or */utils/seed/{length}* is *utils_seed_length*, also HTTP methods may affect class method name like *POST /address* is *address_create* or *DELETE /address* is address_delete. Check out node API documentation at https://nodes.wavesplatform.com/api-docs/index.html

activation_status()

Get node activation status

Returns

address_balance(*address*)

Address waves balance

Parameters **address** –

Returns

Return type *WavesClientResponse*

address_balance_confirmed(*address*, *confirmations*)
Get confirmed address balance

Parameters

- **address** –
- **confirmations** –

Returns

Return type *WavesClientResponse*

address_balance_details(*address*)
Get address details

Parameters **address** –

Returns

Return type *WavesClientResponse*

address_create()
Create node address

Returns

Return type *WavesClientResponse*

address_data(*address_data*)
Set address data

Parameters **address_data** –

Returns

Return type *WavesClientResponse*

address_data_address(*address*, *matches=None*)
Get full address data

Parameters **address** –

Returns

Return type *WavesClientResponse*

address_data_key(*address*, *key*)
Get address data by key

Parameters

- **address** – address in base58
- **key** – string key

Returns

Return type *WavesClientResponse*

address_delete(*address*)
Delete address from node

Parameters **address** – address in base58

Returns

Return type *WavesClientResponse*

address_effective_balance(*address*)
Address effective balance

Parameters **address** –

Returns

Return type *WavesClientResponse*

address_effective_balance_confirmed(*address*, *confirmations*)
Get confirmed address effective balance

Parameters

- **address** –
- **confirmations** –

Returns

Return type *WavesClientResponse*

address_public_key(*public_key*)
Get address by public key

Parameters **public_key** –

Returns

Return type *WavesClientResponse*

address_script_info(*address*)
Get address script info

Parameters **address** – address in base58

Returns

Return type *WavesClientResponse*

address_seed(*address*)
Get address seed

Parameters **address** –

Returns

Return type *WavesClientResponse*

address_sign_text(*address*, *message*)
Sign text

Parameters

- **address** –
- **message** – message string

Returns

Return type *WavesClientResponse*

address_validate(*address*)
Validate address

Parameters **address** –

Returns

Return type *WavesClientResponse*

address_verify_text(*address*, *message_data*)
Verify text

Parameters

- **address** –
- **message_data** –

Returns

Return type *WavesClientResponse*

addresses()
Get node addresses

Returns

Return type *WavesClientResponse*

addresses_sequence(*address_from*, *address_to*)
Get address sequence

Returns

Return type *WavesClientResponse*

alias_broadcast_create(*transaction_data*)
Create alias

Parameters **transaction_data** –

Returns

alias_by_address(*address*)
Get alias by address

Parameters **address** – waves address

Returns

Return type *WavesClientResponse*

alias_by_alias(*alias*)
Get address by alias

Parameters **alias** (*str*) – alias (without preifx and chain id)

Returns

Return type *WavesClientResponse*

asset_broadcast_burn(*transaction_data*)
Burn asset (v1)

Parameters **transaction_data** –

Returns

Return type *WavesClientResponse*

asset_broadcast_issue(*transaction_data*)
Issue asset (v1)

Parameters **transaction_data** –

Returns

Return type *WavesClientResponse*

asset_broadcast_reissue(*transaction_data*)
Reissue asset (transaction v1)

Parameters **transaction_data** –

Returns

Return type *WavesClientResponse*

asset_broadcast_transfer(*transfer_data*)
Broadcast asset transfer transaction (v1)

Parameters **transfer_data** –

Returns

Return type *WavesClientResponse*

asset_distribution_at_height(*asset_id*, *height*, *limit*)
Get asset distribution at height

Parameters

- **asset_id** –
- **height** –
- **limit** –

Returns

Return type *WavesClientResponse*

assets_balance(*address*)
Get assets balance

Parameters **address** –

Returns

Return type *WavesClientResponse*

assets_balance_asset(*address*, *asset_id*)
Get asset balance

Parameters

- **address** –
- **asset_id** –

Returns

Return type *WavesClientResponse*

assets_details(*asset_id*, *full=None*)
Get asset details

Parameters

- **asset_id** –
- **full** –

Returns

Return type *WavesClientResponse*

assets_nft_balance(*address*, *limit*, *after=None*)
Get NFT balance (node version 1.0 and higher)

Parameters

- **address** –
- **limit** –
- **after** –

Returns

Return type *WavesClientResponse*

blocks_address(*address*, *height_from*, *height_to*)
Get blocks generated by address

Parameters

- **address** –
- **height_from** –
- **height_to** –

Returns

Return type *WavesClientResponse*

blocks_at(*height*)
Get block at height

Parameters **height** –

Returns

Return type *WavesClientResponse*

blocks_checkpoint(*checkpoint_data*)
Get blocks checkpoint

Returns

Return type *WavesClientResponse*

blocks_child(*block_signature*)
Get successor of specified block

Parameters **block_signature** –

Returns

Return type *WavesClientResponse*

blocks_delay(*signature*, *block_number*)
Get blocks delay by signature and block number

Parameters

- **signature** –
- **block_number** –

Returns

Return type *WavesClientResponse*

blocks_first()
Get first block

Returns

Return type *WavesClientResponse*

blocks_headers_at(*block_height*)
Get headers of block at height

Parameters **block_height** –

Returns

Return type *WavesClientResponse*

blocks_headers_last()
Last block headers

Returns

Return type *WavesClientResponse*

blocks_headers_sequence(*height_from*, *height_to*)
Get headers of blocks at heights

Parameters

- **height_from** –
- **height_to** –

Returns

Return type *WavesClientResponse*

blocks_height()
Get node blockchain height

Returns

Return type *WavesClientResponse*

blocks_height_signature(*block_signature*)
Get signature of block at height

Parameters **block_signature** –

Returns

Return type *WavesClientResponse*

blocks_last()
Get last block

Returns

Return type *WavesClientResponse*

blocks_signature(*signature*)
Get block by signature

Parameters **signature** –

Returns

Return type *WavesClientResponse*

close_session()
Close requests session

Returns nothing

Return type None

consensus_algo()
Get consensus algorithm

Returns

consensus_base_target()
Get consensus base target

Returns

Return type *WavesClientResponse*

consensus_base_target_block(*block_id*)
Get consensus base target block

Parameters **block_id** –

Returns

consensus_generating_balance_address(*address*)
Get address generating balance

Parameters **address** –

Returns

Return type *WavesClientResponse*

consensus_generation_signature()
Get generation signature of a last block

Returns

Return type *WavesClientResponse*

consensus_generation_signature_block(*signature*, *block_id*)
Get generation signature of a block with specified id

Parameters

- **signature** –
- **block_id** –

Returns

Return type *WavesClientResponse*

leasing_active(*address*)
Get active leasing

Parameters **address** –

Returns

Return type *WavesClientResponse*

leasing_broadcast_cancel_lease(*transaction_data*)
Create cancel lease transaction

Parameters **transaction_data** –

Returns

Return type *WavesClientResponse*

leasing_broadcast_lease(*transaction_data*)
Create lease transaction

Parameters **transaction_data** –

Returns

Return type *WavesClientResponse*

matcher()
Get matcher public key

Returns

Return type *WavesClientResponse*

matcher_balance_reserved(*public_key*)
Get reserved balance of open orders

Parameters **public_key** –

Returns

Return type *WavesClientResponse*

matcher_debug_all_snapshot_offsets()
Get all snapshots' offsets in the queue TODO:

Returns

Return type *WavesClientResponse*

matcher_debug_current_offset()
Get a current offset in the queue TODO:

Returns

Return type *WavesClientResponse*

matcher_debug_last_offset()
Get the last offset in the queue TODO:

Returns

Return type *WavesClientResponse*

matcher_delete_asset_rate(*asset_id*)
Delete rate for the specified asset TODO:

Parameters **asset_id** –

Returns

Return type *WavesClientResponse*

matcher_order_create(*order_data*)
Create order

Returns

Return type *WavesClientResponse*

matcher_order_status(*amount_asset*, *price_asset*, *order_id*)
Get order status for asset pair

Parameters

- **amount_asset** –
- **price_asset** –
- **order_id** –

Returns

Return type *WavesClientResponse*

matcher_orderbook()
Get trading markets

Returns

Return type *WavesClientResponse*

matcher_orderbook_get_asset_pair_status(*amount_asset_id*, *price_asset_id*)
Get orderbook status for asset pair

Parameters

- **amount_asset_id** –
- **price_asset_id** –

Returns

Return type *WavesClientResponse*

matcher_orderbook_history(*public_key*)
Get orderbook history for a public key

Returns

Return type *WavesClientResponse*

matcher_orderbook_remove(*amount_asset_id*, *price_asset_id*)
Remove orderbook for asset pair

Parameters

- **amount_asset_id** –
- **price_asset_id** –

Returns

Return type *WavesClientResponse*

matcher_orderbook_tradable_balance(*amount_asset*, *price_asset*, *address*)
Get tradable balance for asset pair

Parameters

- **amount_asset** –
- **price_asset** –
- **address** –

Returns

Return type *WavesClientResponse*

matcher_orders_address(*address*)
Get address order history for an address

Parameters **address** –

Returns

Return type *WavesClientResponse*

matcher_orders_cancel_order(*order_id*, *transaction_data*)
Cancel order by id

Parameters

- **order_id** –
- **transaction_data** –

Returns

Return type *WavesClientResponse*

matcher_orders_cancel_order_without_signature(*order_id*)
Cancel order with API key

Parameters **order_id** –

Returns

Return type *WavesClientResponse*

matcher_set_asset_rate(*asset_id*, *rate*)
Asset rates TODO:

Parameters

- **asset_id** –
- **rate** –

Returns

Return type *WavesClientResponse*

matcher_settings()
Get matcher settings

Returns

Return type *WavesClientResponse*

matcher_settings_rates()
Get matcher rates in Waves

Returns

Return type *WavesClientResponse*

matcher_transactions_order(*order_id*)
Get exchange transactions created on DEX for the given order

Parameters **order_id** –

Returns

Return type *WavesClientResponse*

matcher_v1_orderbook_get_asset_pair(*amount_asset_id*, *price_asset_id*, *depth=None*)
Get orderbook for asset pair (API v1)

Parameters

- **amount_asset_id** –
- **price_asset_id** –
- **depth** –

Returns

Return type *WavesClientResponse*

node_status()
Get node status

Returns

Return type *WavesClientResponse*

node_stop()
Stop node

Returns

Return type *WavesClientResponse*

node_version()
Get node version

Returns

Return type *WavesClientResponse*

peers_blacklisted()
Get blacklisted peers

Returns

Return type *WavesClientResponse*

peers_clear_blacklist()
Clear peers blacklist

Returns

Return type *WavesClientResponse*

peers_connect(*peer_data*)
Connect to peer

Parameters **peer_data** –

Returns

Return type *WavesClientResponse*

peers_connected()
Get connected peer list

Returns

Return type *WavesClientResponse*

peers_suspended()
Get suspended peers

Returns

Return type *WavesClientResponse*

request(*method*, *endpoint*, *params=None*, *data=None*, *json_data=None*, *headers=None*, *matcher=False*)
Make a request to API

Parameters

- **method** – HTTP method
- **endpoint** – API endpoint
- **params** – query params
- **data** – body data
- **json_data** – body data in json
- **headers** – HTTP headers
- **matcher** – matcher request

Returns handled result if online else request params dict

Return type *WavesClientResponse* or dict

start_session()
Create requests session

Returns nothing

Return type None

transaction_broadcast(*transaction_data*)
Broadcast new transaction

Parameters **transaction_data** –

Returns

Return type *WavesClientResponse*

transaction_calculate_fee(*transaction_data*)
Calculate transaction fee

Parameters **transaction_data** –

Returns

Return type *WavesClientResponse*

transaction_info(*transaction_id*)
Get transaction info

Parameters **transaction_id** –

Returns

Return type *WavesClientResponse*

transaction_sign(*transaction_data*)
Sign transaction

Parameters **transaction_data** –

Returns

Return type *WavesClientResponse*

transaction_sign_address(*signer_address*, *transaction_data*)
Sign address transaction

Parameters

- **signer_address** –
- **transaction_data** –

Returns

Return type *WavesClientResponse*

transaction_unconfirmed()
Get unconfirmed transactions

Returns

Return type *WavesClientResponse*

transaction_unconfirmed_info(*transaction_id*)
Get unconfirmed transaction info

Parameters **transaction_id** –

Returns

Return type *WavesClientResponse*

transaction_unconfirmed_size()
Get size of unconfirmed transactions

Returns

Return type *WavesClientResponse*

transactions_address(*address*, *limit*, *after=None*)
Get address transactions

Parameters

- **address** – Waves address
- **limit** – transaction limit
- **after** – show transactions after transaction id

Returns

Return type *WavesClientResponse*

utils_hash_fast(*message*)
Get FastCryptographicHash for message

Parameters **message** –

Returns

utils_hash_secure(*message*)
Get SecureCryptographicHash for message

Parameters **message** –

Returns

utils_script_compile(*code*)
Compile string code (deprecated on node version 1.0 and higher)

Parameters **code** –

Returns

utils_script_estimate(*code*)
Estimate compiled script

Parameters **code** –

Returns

utils_seed()
Generate random seed on node

Returns

utils_seed_length(*length*)
Generate random seed of specified length on node

Parameters **length** –

Returns

utils_time()
Get node time

Returns

utils_transaction_serialize(*transaction_data*)
Serialize transaction

Parameters **transaction_data** –

Returns

exception wavespy.client.**WavesClientException**
Bases: Exception

Exception for Waves client

class wavespy.client.**WavesClientResponse**(*successful*, *endpoint*, *response_data=None*, *error_code=None*, *error_message=None*)
Bases: object

API client response. Any API method of *WavesClient* returns *WavesClientResponse* with response data or error (if raise_exception is False)

Parameters

- **successful** – is request was successful
- **response_data** – data, returned in response
- **error_code** – error code in response (key "code")
- **error_message** – error message in response (key "message" if response has json else response body as text)

3.3.2 *async_client* module

wavespy.async_client

This module provides async API client class

class wavespy.async_client.**WavesAsyncClient**(*node_address='https://nodes.wavesplatform.com'*, *matcher_address='https://matcher.waves.exchange'*, *chain_id=None*, *api_key=None*, *raise_exception=True*, *request_params=None*, *online=True*)

Bases: *wavespy.client.BaseClient*

Waves async API client class based on *aiohttp.client*

async activation_status()
Get node activation status

Returns

async address_balance(*address*)
Address waves balance

Parameters **address** –

Returns

Return type *WavesAsyncClientResponse*

async address_balance_confirmed(*address*, *confirmations*)
Get confirmed address balance

Parameters

- **address** –
- **confirmations** –

Returns

Return type *WavesAsyncClientResponse*

async address_balance_details(*address*)
Get address details

Parameters **address** –

Returns

Return type *WavesAsyncClientResponse*

async address_create()
Create node address

Returns

Return type *WavesAsyncClientResponse*

async address_data(*address_data*)
Set address data

Parameters **address_data** –

Returns

Return type *WavesAsyncClientResponse*

async address_data_address(*address*, *matches=None*)
Get full address data

Parameters **address** –

Returns

Return type *WavesAsyncClientResponse*

async address_data_key(*address*, *key*)
Get address data by key

Parameters

- **address** – address in base58
- **key** – string key

Returns

Return type *WavesAsyncClientResponse*

async address_delete(*address*)
Delete address from node

Parameters **address** – address in base58

Returns

Return type *WavesAsyncClientResponse*

async address_effective_balance(*address*)
Address effective balance

Parameters **address** –

Returns

Return type *WavesAsyncClientResponse*

async address_effective_balance_confirmed(*address*, *confirmations*)
Get confirmed address effective balance

Parameters

- **address** –
- **confirmations** –

Returns

Return type *WavesAsyncClientResponse*

async address_public_key(*public_key*)
Get address by public key

Parameters **public_key** –

Returns

Return type *WavesAsyncClientResponse*

async address_script_info(*address*)
Get address script info

Parameters **address** – address in base58

Returns

Return type *WavesAsyncClientResponse*

async address_seed(*address*)
Get address seed

Parameters **address** –

Returns

Return type *WavesAsyncClientResponse*

async address_sign_text(*address*, *message*)
Sign text

Parameters

- **address** –
- **message** – message string

Returns

Return type *WavesAsyncClientResponse*

async address_validate(*address*)
Validate address

Parameters **address** –

Returns

Return type *WavesAsyncClientResponse*

async address_verify_text(*address*, *message_data*)
Verify text

Parameters

- **address** –
- **message_data** –

Returns

Return type *WavesAsyncClientResponse*

async addresses()
Get node addresses

Returns

Return type *WavesAsyncClientResponse*

async addresses_sequence(*address_from*, *address_to*)
Get address sequence

Returns

Return type *WavesAsyncClientResponse*

async alias_broadcast_create(*transaction_data*)
Create alias

Parameters **transaction_data** –

Returns

async alias_by_address(*address*)
Get alias by address

Parameters **address** – waves address

Returns

Return type *WavesAsyncClientResponse*

async alias_by_alias(*alias*)
Get address by alias

Parameters **alias** (*str*) – alias (without preifx and chain id)

Returns

Return type *WavesAsyncClientResponse*

async asset_broadcast_burn(*transaction_data*)
Burn asset (v1)

Parameters **transaction_data** –

Returns

Return type *WavesAsyncClientResponse*

async asset_broadcast_issue(*transaction_data*)
Issue asset (v1)

Parameters **transaction_data** –

Returns

Return type *WavesAsyncClientResponse*

async asset_broadcast_reissue(*transaction_data*)
Reissue asset (transaction v1)

Parameters **transaction_data** –

Returns

Return type *WavesAsyncClientResponse*

async asset_broadcast_transfer(*transfer_data*)
Broadcast asset transfer transaction (v1)

Parameters **transfer_data** –

Returns

Return type *WavesAsyncClientResponse*

async asset_distribution_at_height(*asset_id*, *height*, *limit*)
Get asset distribution at height

Parameters

- **asset_id** –
- **height** –
- **limit** –

Returns

Return type *WavesAsyncClientResponse*

async assets_balance(*address*)
Get assets balance

Parameters **address** –

Returns

Return type *WavesAsyncClientResponse*

async assets_balance_asset(*address*, *asset_id*)
Get asset balance

Parameters

- **address** –
- **asset_id** –

Returns

Return type *WavesAsyncClientResponse*

async assets_details(*asset_id*, *full=None*)
Get asset details

Parameters

- **asset_id** –
- **full** –

Returns

Return type *WavesAsyncClientResponse*

async assets_nft_balance(*address*, *limit*, *after=None*)
Get NFT balance (node version 1.0 and higher)

Parameters

- **address** –
- **limit** –
- **after** –

Returns

Return type *WavesAsyncClientResponse*

async blocks_address(*address*, *height_from*, *height_to*)
Get blocks generated by address

Parameters

- **address** –
- **height_from** –
- **height_to** –

Returns

Return type *WavesAsyncClientResponse*

async blocks_at(*height*)
Get block at height

Parameters **height** –

Returns

Return type *WavesAsyncClientResponse*

async blocks_checkpoint(*checkpoint_data*)
Get blocks checkpoint

Returns

Return type *WavesAsyncClientResponse*

async blocks_child(*block_signature*)
Get successor of specified block

Parameters **block_signature** –

Returns

Return type *WavesAsyncClientResponse*

async blocks_delay(*signature*, *block_number*)
Get blocks delay by signature and block number

Parameters

- **signature** –
- **block_number** –

Returns

Return type *WavesAsyncClientResponse*

async blocks_first()
Get first block

Returns

Return type *WavesAsyncClientResponse*

async blocks_headers_at(*block_height*)
Get headers of block at height

Parameters **block_height** –

Returns

Return type *WavesAsyncClientResponse*

async blocks_headers_last()
Last block headers

Returns

Return type *WavesAsyncClientResponse*

async blocks_headers_sequence(*height_from*, *height_to*)
Get headers of blocks at heights

Parameters

- **height_from** –
- **height_to** –

Returns

Return type *WavesAsyncClientResponse*

async blocks_height()
Get node blockchain height

Returns

Return type *WavesAsyncClientResponse*

async blocks_height_signature(*block_signature*)
Get signature of block at height

Parameters **block_signature** –

Returns

Return type *WavesAsyncClientResponse*

async blocks_last()
Get last block

Returns

Return type *WavesAsyncClientResponse*

async blocks_signature(*signature*)
Get block by signature

Parameters **signature** –

Returns

Return type *WavesAsyncClientResponse*

async close_session()
Close aiothttp client session

Returns nothing

Return type None

async consensus_algo()
Get consensus algorithm

Returns

async consensus_base_target()
Get consensus base target

Returns

Return type *WavesAsyncClientResponse*

async consensus_base_target_block(*block_id*)
Get consensus base target block

Parameters **block_id** –

Returns

async consensus_generating_balance_address(*address*)
Get address generating balance

Parameters **address** –

Returns

Return type *WavesAsyncClientResponse*

async consensus_generation_signature()
Get generation signature of a last block

Returns

Return type *WavesAsyncClientResponse*

async consensus_generation_signature_block(*signature*, *block_id*)
Get generation signature of a block with specified id

Parameters

- **signature** –
- **block_id** –

Returns

Return type *WavesAsyncClientResponse*

async leasing_active(*address*)
Get active leasing

Parameters **address** –

Returns

Return type *WavesAsyncClientResponse*

async leasing_broadcast_cancel_lease(*transaction_data*)
Create cancel lease transaction

Parameters **transaction_data** –

Returns

Return type *WavesAsyncClientResponse*

async leasing_broadcast_lease(*transaction_data*)
Create lease transaction

Parameters **transaction_data** –

Returns

Return type *WavesAsyncClientResponse*

async matcher()
Get matcher public key

Returns

Return type *WavesAsyncClientResponse*

async matcher_balance_reserved(*public_key*)
Get reserved balance of open orders

Parameters **public_key** –

Returns

Return type *WavesAsyncClientResponse*

async matcher_order_create(*order_data*)
Create order

Returns

Return type *WavesAsyncClientResponse*

async matcher_order_status(*amount_asset*, *price_asset*, *order_id*)
Get order status for asset pair

Parameters

- **amount_asset** –
- **price_asset** –
- **order_id** –

Returns

Return type *WavesAsyncClientResponse*

async matcher_orderbook()
Get trading markets

Returns

Return type *WavesAsyncClientResponse*

async matcher_orderbook_get_asset_pair(*amount_asset_id*, *price_asset_id*)
Get orderbook for asset pair

Parameters

- **amount_asset_id** –
- **price_asset_id** –

Returns

Return type *WavesAsyncClientResponse*

async matcher_orderbook_get_asset_pair_status(*amount_asset_id*, *price_asset_id*)
Get orderbook status for asset pair

Parameters

- **amount_asset_id** –
- **price_asset_id** –

Returns

Return type *WavesAsyncClientResponse*

async matcher_orderbook_history(*public_key*)
Get orderbook history for a public key

Returns

Return type *WavesAsyncClientResponse*

async matcher_orderbook_remove(*amount_asset_id*, *price_asset_id*)
Remove orderbook for asset pair

Parameters

- **amount_asset_id** –
- **price_asset_id** –

Returns

Return type *WavesAsyncClientResponse*

async matcher_orderbook_tradable_balance(*amount_asset*, *price_asset*, *address*)
Get tradable balance for asset pair

Parameters

- **amount_asset** –
- **price_asset** –
- **address** –

Returns

Return type *WavesAsyncClientResponse*

async matcher_orders_address(*address*)
Get address order history for an address

Parameters **address** –

Returns

Return type *WavesAsyncClientResponse*

async matcher_orders_cancel_order(*order_id*, *transaction_data*)
Cancel order by id

Parameters **order_id** –

Returns

Return type *WavesAsyncClientResponse*

async matcher_settings()
Get matcher settings

Returns

Return type *WavesAsyncClientResponse*

async matcher_transactions_order(*order_id*)
Get exchange transactions created on DEX for the given order

Parameters **order_id** –

Returns

Return type *WavesAsyncClientResponse*

async node_status()
Get node status

Returns

Return type *WavesAsyncClientResponse*

async node_stop()
Stop node

Returns

Return type *WavesAsyncClientResponse*

async node_version()
Get node version

Returns

Return type *WavesAsyncClientResponse*

async peers_blacklisted()
Get blacklisted peers

Returns

Return type *WavesAsyncClientResponse*

async peers_clear_blacklist()
Clear peers blacklist

Returns

Return type *WavesAsyncClientResponse*

async peers_connect(*peer_data*)
Connect to peer

Parameters **peer_data** –

Returns

Return type *WavesAsyncClientResponse*

async peers_connected()
Get connected peer list

Returns

Return type *WavesAsyncClientResponse*

async peers_suspended()
Get suspended peers

Returns

Return type *WavesAsyncClientResponse*

async request(*method*, *endpoint*, *params=None*, *data=None*, *json_data=None*, *headers=None*, *matcher=False*)
Make a asynchronous request to API If session was created outside (e.g. in context manager method) don't create new and don't close on request finish, else create new session and close it after request

Parameters

- **method** – HTTP method
- **endpoint** – API endpoint
- **params** – query params
- **data** – body data
- **json_data** – body data in json
- **headers** – HTTP headers

Param matcher: matcher request

Returns handled result if online else request params dict

Return type *WavesAsyncClientResponse* or dict

async start_session()
Create aiothttp client session

Returns nothing

Return type None

async transaction_broadcast(*transaction_data*)
Broadcast new transaction

Parameters **transaction_data** –

Returns

Return type *WavesAsyncClientResponse*

async transaction_calculate_fee(*transaction_data*)
Calculate transaction fee

Parameters **transaction_data** –

Returns

Return type *WavesAsyncClientResponse*

async transaction_info(*transaction_id*)
Get transaction info

Parameters **transaction_id** –

Returns

Return type *WavesAsyncClientResponse*

async transaction_sign(*transaction_data*)
Sign transaction

Parameters **transaction_data** –

Returns

Return type *WavesAsyncClientResponse*

async transaction_sign_address(*signer_address*, *transaction_data*)
Sign address transaction

Parameters

- **signer_address** –
- **transaction_data** –

Returns

Return type *WavesAsyncClientResponse*

async transaction_unconfirmed()
Get unconfirmed transactions

Returns

Return type *WavesAsyncClientResponse*

async transaction_unconfirmed_info(*transaction_id*)
Get unconfirmed transaction info

Parameters **transaction_id** –

Returns

Return type *WavesAsyncClientResponse*

async transaction_unconfirmed_size()
Get size of unconfirmed transactions

Returns

Return type *WavesAsyncClientResponse*

async transactions_address(*address*, *limit*, *after=None*)
Get address transactions

Parameters

- **address** – Waves address
- **limit** – transaction limit

- **after** – show transactions after transaction id

Returns

Return type *WavesAsyncClientResponse*

async utils_hash_fast(*message*)
Get FastCryptographicHash for message

Parameters **message** –

Returns

async utils_hash_secure(*message*)
Get SecureCryptographicHash for message

Parameters **message** –

Returns

async utils_script_compile(*code*)
Compile string code (deprecated on node version 1.0 and higher)

Parameters **code** –

Returns

async utils_script_estimate(*code*)
Estimate compiled script

Parameters **code** –

Returns

async utils_seed()
Generate random seed on node :return:

async utils_seed_length(*length*)
Generate random seed of specified length on node

Parameters **length** –

Returns

async utils_time()
Get node time

Returns

async utils_transaction_serialize(*transaction_data*)
Serialize transaction

Parameters **transaction_data** –

Returns

exception wavespy.async_client.**WavesAsyncClientException**
Bases: *wavespy.client.WavesClientException*

Exception for async waves client

class wavespy.async_client.**WavesAsyncClientResponse**(*successful*, *endpoint*, *response_data=None*, *error_code=None*, *error_message=None*)
Bases: *wavespy.client.WavesClientResponse*

Async API client response

3.3.3 *utils* module

wavespy.utils

Utilities for addresses and transactions

wavespy.utils.address

Waves address

class wavespy.utils.address.**BaseWavesAddress**(*value=None*, *private_key=None*, *public_key=None*, *address_seed=None*, *chain_id=None*, *nonce=0*, *node_address='https://nodes.wavesplatform.com'*, *version=1*, *online=True*, *client_request_params=None*)

Bases: object

Base address class. Has methods for transaction data generation, common for child address classes

Parameters

- **value** – address text value in base58
- **private_key** – address private key string in base58
- **public_key** – address public key string in base58
- **address_seed** – address seed
- **chain_id** – address chain id
- **nonce** – nonce
- **node_address** – node API address for data retrieve and transaction broadcast
- **version** – address version (currently, only for information)
- **online** – make requests or return only request data
- **client_request_params** – client API request param (check API client doc)

property base58_seed

Seed encoded base58

Returns seed in base58

Return type bytes

property can_sign

Ability to sign requests

Returns yes or no

Return type bool

property chain

Address chain name

Returns chain name

Return type str

property client

Current API client instance

Returns API client instance

load_from_json(*file_path*, *decode_seed=False*)
Get address data from json file :param file_path: file path :param decode_seed: decode seed from base58 :return: nothing :rtype: None

save_as_json(*file_path*, *encode_seed=False*)
Save address data as json :param file_path: file path :param encode_seed: encode seed to base58 :return: nothing :rtype: None

class wavespy.utils.address.**WavesAddress**(**args*, ***kwargs*)
Bases: *wavespy.utils.address.BaseWavesAddress*

Waves address class. Simplifies transaction data generation and data requests

Parameters

- **value** – address text value in base58
- **private_key** – address private key string in base58
- **public_key** – address public key string in base58
- **address_seed** – address seed
- **chain_id** – address chain id
- **nonce** – nonce

asset_sponsorship(*asset_id*, *min_sponsored_asset_fee*, *transaction_fee=100000000*, *timestamp=0*)
Sponsor asset

Parameters

- **asset_id** –
- **min_sponsored_asset_fee** –
- **transaction_fee** –
- **timestamp** –

Returns

burn_asset(*asset_id*, *quantity*, *transaction_fee=100000*, *timestamp=0*)
Burn asset

Parameters

- **asset_id** – asset id
- **quantity** – asset quantity
- **transaction_fee** –
- **timestamp** – transaction timestamp

Returns

create_alias(*alias*, *transaction_fee=100000*, *timestamp=0*)
Create alias for address

Parameters

- **alias** – alias (min 4, max 30 chars)
- **transaction_fee** –

- **timestamp** –

Returns

data_transaction(*data*, *fee=0*, *version=1*, *timestamp=0*)
Create data transaction

Parameters

- **data** (*list*) – data for data transaction (list of dicts with keys type, key, value) Suitable python types for transaction data types: - boolean - *bool* - binary - *bytes* (converts them to base64 string) - integer - *int* - string - *str*
- **version** (*int*) – data transaction version
- **fee** (*int*) – transaction fee. 1000000 is minimum.
- **timestamp** (*int*) – transaction timestamp

Returns client request result

Return type *WavesClientResponse* or dict

from_alias(*alias*)
Set address value from alias

Parameters **alias** – alias of address

Returns address object with address value

get_address_data()
Get address data from blockchain

Returns data in dict

Return type dict

get_address_info()
Collect address info (balances, assets, data)

Returns address info dict

Return type dict

get_aliases(*flat=True*)
Get address aliases

Parameters **flat** – get only alias names without chain ids

Returns list of dicts with prefix, chain ids and alias names or only alias names

Return type list or dict

get_assets()
Get address assets

Returns asset balances in dict

Return type int or dict

get_balance()
Get address balance

Returns balance value

Return type int or dict

get_confirmed_balance(*confirmations*)
Get address balance

Returns balance value

Return type int or dict

get_effective_balance()
Get address effective balance

Returns balance value

Return type int or dict

issue_asset(*name*, *description*, *quantity*, *decimals*, *reissuable*, *transaction_fee=100000000*, *version=1*, *timestamp=0*)
Issue asset in waves blockchain

Parameters

- **name** – asset name (min 4, max 16 chars)
- **description** – asset description
- **quantity** – asset quantity
- **decimals** – asset decimals
- **reissuable** – is asset reissuable
- **transaction_fee** – asset issue transaction fee
- **version** – transaction version
- **timestamp** – transaction timestamp

Returns

issue_smart_asset(*name*, *description*, *quantity*, *decimals*, *reissuable*, *script*, *transaction_fee=100000000*, *timestamp=0*)
Issue smart asset in waves blockchain (asset with script)

Parameters

- **name** – asset name (min 4, max 16 chars)
- **description** – asset description
- **quantity** – asset quantity
- **decimals** – asset decimals
- **reissuable** – is asset reissuable
- **transaction_fee** – asset issue transaction fee
- **timestamp** – transaction timestamp

Returns

lease_cancel(*transaction_id*, *transaction_fee=100000*, *timestamp=0*)
Cancel waves lease

Parameters

- **transaction_id** –
- **transaction_fee** –
- **timestamp** –

Returns

lease_waves(*recipient*, *amount*, *transaction_fee=100000*, *timestamp=0*)
Lease waves to address

Parameters

- **recipient** –
- **amount** –
- **transaction_fee** –
- **timestamp** –

Returns

mass_transfer_assets(*transfer_data*, *asset_id=None*, *attachment=None*, *version=1*, *timestamp=None*)
Mass transfer waves

Parameters

- **transfer_data** – list of dicts with recipient and amount i.e. *[{ 'recipient': '3N1xca2DY8AEwqRDAJpzUgY99eq8J9h4rB3', 'amount': 1000 }]*
- **attachment** – transaction attachment
- **asset_id** – ID of transferring asset
- **timestamp** – transaction timestamp

Returns

mass_transfer_waves(*transfer_data*, *attachment=None*, *timestamp=None*)
Mass transfer waves

Parameters

- **transfer_data** – list of dicts with recipient and amount i.e. *[{ 'recipient': '3N1xca2DY8AEwqRDAJpzUgY99eq8J9h4rB3', 'amount': 1000 }]*
- **attachment** – transaction attachment
- **transaction_fee** – mass transfer transaction fee
- **timestamp** – transaction timestamp

Returns

Return type *WavesClientResponse* or dict

reissue_asset(*asset_id*, *quantity*, *reissuable*, *transaction_fee=100000000*, *timestamp=0*)
Reissue asset in waves blockchain

Parameters

- **asset_id** – asset id
- **quantity** – asset quantity
- **transaction_fee** –
- **timestamp** – transaction timestamp

Returns

set_asset_script(*script*, *asset_id*, *transaction_fee=100000000*, *timestamp=None*, *version=1*)
Set script for asset

Parameters

- **script** –
- **asset_id** –
- **transaction_fee** –
- **timestamp** –
- **version** –

Returns

set_script(*script*, *transaction_fee=1000000*, *timestamp=None*, *version=1*)
Set script for account

Parameters

- **script** –
- **transaction_fee** –
- **timestamp** –
- **version** –

Returns

transfer_asset(*recipient*, *asset_id*, *fee_asset_id*, *amount*, *attachment=None*, *transaction_fee=100000*, *timestamp=0*)
Send asset to address. If asset_id or fee_asset_id is None then waves will be used

Parameters

- **recipient** (*str or WavesAddress or WavesAsyncAddress*) – recipient address in base58
- **asset_id** (*str or None*) – asset id
- **fee_asset_id** (*str or None*) – fee asset id
- **amount** (*int*) – amount of waves
- **attachment** (*str*) – attachment string
- **transaction_fee** (*int*) – fee for transfer transaction
- **timestamp** (*int*) – timestamp of transaction

Returns transfer result

Return type *WavesClientResponse* or dict

transfer_waves(*recipient*, *amount*, *attachment=None*, *transaction_fee=100000*, *timestamp=0*)
Send waves to address

Parameters

- **recipient** (*str or WavesAddress or WavesAsyncAddress*) – recipient address in base58
- **amount** (*int*) – amount of waves
- **attachment** (*str*) – attachment string
- **transaction_fee** (*int*) – fee for transfer transaction
- **timestamp** (*int*) – timestamp of transaction

Returns transfer result

Return type *WavesClientResponse* or dict

validate()
Validate address :return: :rtype bool or dict

wavespy.utils.address.**sign_required**(*method*)
Decorator. Check if address can sign request data, if not then raises ValueError

Parameters **method** – address class method

Returns wrapped method

Return type Callable

Raises ValueError

wavespy.utils.address_generator

Address generator for Waves blockchain

class wavespy.utils.address_generator.**WavesAddressGenerator**(*node_address='https://nodes.wavesplatform.*
async_address=False)

Bases: object

Waves address generator

Parameters

- **node_address** (*str*) – node API URL
- **async_address** (*bool*) – return address with async client instead of default sync

generate(*value=None*, *private_key=None*, *public_key=None*, *seed=None*, *chain_id='W'*, *nonce=0*, *version=1*, *online=True*, *client_request_params=None*)

Generate address

- if there is no seed and private key is specified then generate address value and public key
- if there is no seed and public key is specified then generate address value
- if there is no seed and value is specified with any key then validate them and return address object
- if there is no seed and no any key but value is specified validation or generation is not performed, address object would be returned with the same value and chain_id

Parameters

- **value** – address string value in base58
- **private_key** – private key string value in base58
- **public_key** – private key string value in base58
- **seed** – seed
- **chain_id** – chain id value
- **nonce** – nonce
- **version** – address version
- **online** – if True send requests else return request params dict

Param client_request_params:

Returns WavesAddress or AsyncWavesAddress object with different attributes

Return type *WavesAddress* or *WavesAsyncAddress*

generate_from_private_key(*private_key*, *chain_id*, *version*)
Generate address from private key (address value, public key)

Parameters

- **private_key** –
- **chain_id** –

Returns data for address object creation

Return type dict

generate_from_public_key(*public_key*, *chain_id*, *version*)
Generate address from public key (address value only)

Parameters

- **public_key** – public key in base58
- **chain_id** – chain id

Returns data for address object creation

Return type dict

classmethod generate_from_seed(*seed*, *chain_id*, *nonce*, *version*)
Generate address from seed (address value, private and public keys)

Returns data for address object creation

Return type dict

classmethod generate_private_key(*seed*, *nonce=0*)
Generate private key from seed

Parameters

- **seed** – seed value
- **nonce** – nonce value

Returns

static generate_seed(*language=None*, *strength=None*)
Generate seed

Returns seed string

Return type str

validate_address(*value*, *private_key=None*, *public_key=None*, *chain_id='W'*, *version=None*)
Validate address. If address data is valid returns it else raises error

Parameters

- **value** – address value
- **chain_id** – Waves chain id
- **private_key** – address private key
- **public_key** – address public key
- **version** – address version

Returns data for address object creation

Return type dict

Raises ValueError

wavespy.utils.async_address

Waves address with async methods

class wavespy.utils.async_address.**WavesAsyncAddress**(**args*, ***kwargs*)
Bases: *wavespy.utils.address.BaseWavesAddress*

Waves address with async client. Object methods will return coroutines, so you should run them in event loop

asset_sponsorship(*asset_id*, *min_sponsored_asset_fee*, *transaction_fee=100000000*, *timestamp=0*)
Sponsor asset

Parameters

- **asset_id** –
- **min_sponsored_asset_fee** –
- **transaction_fee** –
- **timestamp** –

Returns

burn_asset(*asset_id*, *quantity*, *transaction_fee=100000*, *timestamp=0*)
Burn asset

Parameters

- **asset_id** – asset id
- **quantity** – asset quantity
- **transaction_fee** –
- **timestamp** – transaction timestamp

Returns

create_alias(*alias*, *transaction_fee=100000*, *timestamp=0*)
Create alias for address

Parameters

- **alias** – alias (min 4, max 30 chars)
- **transaction_fee** –
- **timestamp** –

Returns

data_transaction(*data*, *version=1*, *timestamp=0*)
Create data transaction

Parameters

- **data** (*list*) – data for data transaction (list of dicts with keys type, key, value) Suitable python types for transaction data types: - boolean - *bool* - binary - *bytes* (converts them to base64 string) - integer - *int* - string - *str*

- **version** (*int*) – data transaction version
- **timestamp** (*int*) – transaction timestamp

Returns client request result

Return type *WavesAsyncClientResponse* or dict

async from_alias(*alias*)
Set address value from alias

Parameters **alias** – alias of address

Returns address object with address value

async get_address_data()
Get address data from blockchain

Returns data in dict

Return type dict

async get_address_info()
Collect address info (balances, assets, data)

Returns address info dict

Return type dict

async get_aliases(*flat=True*)
Get address aliases

Parameters **flat** – get only alias names without chain ids

Returns list of dicts with prefix, chain ids and alias names or only alias names

Return type list

async get_assets()
Get address assets

Returns asset balances in dict

Return type dict

async get_balance()
Get address balance

Returns balance value

Return type int

async get_confirmed_balance(*confirmations*)
Get address balance

Returns balance value

Return type int

async get_effective_balance()
Get address effective balance

Returns balance value

Return type int

issue_asset(*name*, *description*, *quantity*, *decimals*, *reissuable*, *transaction_fee=100000000*, *version=1*, *timestamp=0*)
Issue asset in waves blockchain

Parameters

- **name** – asset name (min 4, max 16 chars)
- **description** – asset description
- **quantity** – asset quantity
- **decimals** – asset decimals
- **reissuable** – is asset reissuable
- **transaction_fee** – asset issue transaction fee
- **version** – transaction version
- **timestamp** – transaction timestamp

Returns

issue_smart_asset(*name*, *description*, *quantity*, *decimals*, *reissuable*, *script*, *transaction_fee=100000000*, *timestamp=0*)

Issue smart asset in waves blockchain (asset with script)

Parameters

- **name** – asset name (min 4, max 16 chars)
- **description** – asset description
- **quantity** – asset quantity
- **decimals** – asset decimals
- **reissuable** – is asset reissuable
- **transaction_fee** – asset issue transaction fee
- **timestamp** – transaction timestamp

Returns

lease_cancel(*transaction_id*, *transaction_fee=100000*, *timestamp=0*)

Cancel waves lease

Parameters

- **transaction_id** –
- **transaction_fee** –
- **timestamp** –

Returns

lease_waves(*recipient*, *amount*, *transaction_fee=100000*, *timestamp=0*)

Lease waves to address

Parameters

- **recipient** –
- **amount** –
- **transaction_fee** –
- **timestamp** –

Returns

mass_transfer_assets(*transfer_data*, *asset_id=None*, *attachment=None*, *version=1*, *timestamp=None*)
Mass transfer waves

Parameters

- **transfer_data** – list of dicts with recipient and amount i.e. *[{ 'recipient': '3N1xca2DY8AEwqRDAJpzUgY99eq8J9h4rB3', 'amount': 1000 }]*
- **attachment** – transaction attachment
- **asset_id** – ID of transferring asset
- **timestamp** – transaction timestamp

Returns

mass_transfer_waves(*transfer_data*, *attachment=None*, *timestamp=None*)
Mass transfer waves

Parameters

- **transfer_data** – list of dicts with recipient and amount i.e. *[{ 'recipient': '3N1xca2DY8AEwqRDAJpzUgY99eq8J9h4rB3', 'amount': 1000 }]*
- **attachment** – transaction attachment
- **transaction_fee** – mass transfer transaction fee
- **timestamp** – transaction timestamp

Returns

Return type *WavesAsyncClientResponse* or dict

reissue_asset(*asset_id*, *quantity*, *reissuable*, *transaction_fee=100000000*, *timestamp=0*)
Reissue asset in waves blockchain

Parameters

- **asset_id** – asset id
- **quantity** – asset quantity
- **transaction_fee** –
- **timestamp** – transaction timestamp

Returns

set_asset_script(*script*, *asset_id*, *transaction_fee=100000000*, *timestamp=None*, *version=1*)
Set script for asset

Parameters

- **script** –
- **asset_id** –
- **transaction_fee** –
- **timestamp** –
- **version** –

Returns

set_script(*script*, *transaction_fee=1000000*, *timestamp=None*, *version=1*)
Set script for account

Parameters

- **script** –
- **transaction_fee** –
- **timestamp** –
- **version** –

Returns

transfer_asset(*recipient*, *asset_id*, *fee_asset_id*, *amount*, *attachment=None*, *transaction_fee=100000*, *timestamp=0*)

Send asset to address. If asset_id or fee_asset_id is None then waves will be used

Parameters

- **recipient** (*str or WavesAddress or WavesAsyncAddress*) – recipient address in base58
- **asset_id** (*str or None*) – asset id
- **fee_asset_id** (*str or None*) – fee asset id
- **amount** (*int*) – amount of waves
- **attachment** (*str*) – attachment string
- **transaction_fee** (*int*) – fee for transfer transaction
- **timestamp** (*int*) – timestamp of transaction

Returns transfer result

Return type *WavesAsyncClientResponse* or dict

transfer_waves(*recipient*, *amount*, *attachment=None*, *transaction_fee=100000*, *timestamp=0*)

Send waves to address

Parameters

- **recipient** (*str or WavesAddress or WavesAsyncAddress*) – recipient address in base58
- **amount** (*int*) – amount of waves
- **attachment** (*str*) – attachment string
- **transaction_fee** (*int*) – fee for transfer transaction
- **timestamp** (*int*) – timestamp of transaction

Returns transfer result

Return type *WavesAsyncClientResponse* or dict

async validate()

Validate address

Returns validation result

Return type bool

wavespy.utils.crypto

Cryptographic functions

wavespy.utils.crypto.**sign_with_private_key**(*private_key*, *data*)
Sign data with private key

Parameters

- **private_key** (*str*) – private key value in base58
- **data** (*bytes*) – data to sign

Returns signed data

Return type bytes

wavespy.utils.crypto.**verify_signature**(*public_key*, *signature*, *data*)
Verify data signature

Parameters

- **public_key** (*bytes*) – public key value
- **signature** – signature value
- **data** – signed data

Returns valid or not

Return type bool

CHAPTER

FOUR

INDICES AND TABLES

- genindex
- modindex
- search

PYTHON MODULE INDEX

W

INDEX

A

B

C

D

F

G

I

L

M

N

P

R

S

T

U

V

W