hiveengine Documentation

Release 0.2.3

holger80

Jul 22, 2020

Contents

CHAPTER 1

General

1.1 Installation

The minimal working python version is 3.5.x

Install beem with pip:

```
pip install -U hiveengine
```

Sometimes this does not work. Please try:

```
pip3 install -U hiveengine
```

or:

```
python -m pip install hiveengine
```

1.1.1 Manual installation

You can install beem from this repository if you want the latest but possibly non-compiling version:

```
git clone https://github.com/holgern/hiveengine.git
cd hiveengine
python setup.py build

python setup.py install --user
```

Run tests after install:

```
pytest
```

1.2 Quickstart

1.3 hiveengine CLI

1.3.1 Commands

hiveengine

```
hiveengine [OPTIONS] COMMAND1 [ARGS]... [COMMAND2 [ARGS]...]...
```

Options

-d, --no-broadcast
Do not broadcast

-v, --verbose <verbose>
Verbosity

--version
Show the version and exit.

balance

Show token balance and value

```
hiveengine balance [OPTIONS]
```

Options

-a, --account <account>
Buy with this account (defaults to “default_account”)

buy

Put a buy-order for a token to the hive-engine market

```
hiveengine buy [OPTIONS] AMOUNT TOKEN PRICE
```

Options

-a, --account <account>
Buy with this account (defaults to “default_account”)

Arguments

AMOUNT
Required argument

TOKEN
Required argument

PRICE
Required argument

buybook

Returns the buy book for the given token

```
hiveengine buybook [OPTIONS] [TOKEN]
```

Options

-a, --account <account>
Buy with this account (defaults to "default_account")

Arguments

TOKEN
Optional argument

cancel

Cancel a buy/sell order

order_type is either sell or buy

```
hiveengine cancel [OPTIONS] ORDER_TYPE [ORDER_ID]
```

Options

-a, --account <account>
Buy with this account (defaults to "default_account")

-y, --yes
Answer yes to all questions

Arguments

ORDER_TYPE
Required argument

ORDER_ID
Optional argument

cancel-unstake

unstake a token

```
hiveengine cancel-unstake [OPTIONS] TRX_ID
```

Options

-a, --account <account>
Transfer from this account

Arguments

TRX_ID
Required argument

collection

Return NFT collection for an account

```
hiveengine collection [OPTIONS] ACCOUNT [SYMBOL]...
```

Options

-s, --sort-by-id
Sort NFTs by their ID

Arguments

ACCOUNT
Required argument

SYMBOL
Optional argument(s)

deposit

Deposit HIVE to market in exchange for SWAP.HIVE.

```
hiveengine deposit [OPTIONS] AMOUNT
```

Options

-a, --account <account>
withdraw from this account

Arguments

AMOUNT
Required argument

info

Show basic blockchain info

General information about hive-engine, a block, an account, a token, and a transaction id

```
hiveengine info [OPTIONS] [OBJECTS]...
```

Arguments

OBJECTS
Optional argument(s)

issue

Issue a token

```
hiveengine issue [OPTIONS] TO AMOUNT TOKEN
```

Options

-a, --account <account>
Transfer from this account

Arguments

TO
Required argument

AMOUNT
Required argument

TOKEN
Required argument

nft

Returns information about an NFT ID

```
hiveengine nft [OPTIONS] SYMBOL [NFTID]...
```

Arguments

SYMBOL
Required argument

NFTID
Optional argument(s)

nftbuy

Buy nfts from the market

```
hiveengine nftbuy [OPTIONS] SYMBOL [NFT_IDS]...
```

Options

-a, --account <account>
Buy with this account (defaults to “default_account”)

-m, --market_account <market_account>
Market account which will receive the fee (defaults to “nftmarket”)

-y, --yes
Answer yes to all questions

Arguments

SYMBOL
Required argument

NFT_IDS
Optional argument(s)

nftcancel

Cancel a nft sell order

```
hiveengine nftcancel [OPTIONS] SYMBOL [NFT_IDS]...
```

Options

-a, --account <account>
Buy with this account (defaults to “default_account”)

-y, --yes
Answer yes to all questions

Arguments

SYMBOL
Required argument

NFT_IDS
Optional argument(s)

nftchangeprice

Cancel a nft sell order

```
hiveengine nftchangeprice [OPTIONS] SYMBOL [NFT_IDS]... NEWPRICE
```

Options

-a, --account <account>
Buy with this account (defaults to "default_account")

-y, --yes
Answer yes to all questions

Arguments

SYMBOL
Required argument

NFT_IDS
Optional argument(s)

NEWPRICE
Required argument

nftinfo

Returns information about an NFT symbol

```
hiveengine nftinfo [OPTIONS] [SYMBOL]...
```

Arguments

SYMBOL
Optional argument(s)

nftlist

Show list of all NFTs

```
hiveengine nftlist [OPTIONS]
```

nftopen

Returns the open interest book for the given symbol

```
hiveengine nftopen [OPTIONS] SYMBOL
```

Options

-g, --grouping <grouping>
Can be set to a grouping parameter, or to parameter.value

-v, --value <value>
Set property value, can be used when grouping is set to a property parameter

-s, --price-symbol <price_symbol>
Limit to this price symbol

Arguments

SYMBOL
Required argument

nftparams

Show params of all NFTs

```
hiveengine nftparams [OPTIONS]
```

nftsell

Create a sell order on the market

```
hiveengine nftsell [OPTIONS] SYMBOL [NFT_IDS]... PRICE PRICE_SYMBOL
```

Options

-a, --account <account>
Buy with this account (uses the beem default account when not set)

-f, --fee <fee>
Market fee 500 -> 5% (defaults is 500)

-y, --yes
Answer yes to all questions

Arguments

SYMBOL
Required argument

NFT_IDS
Optional argument(s)

PRICE
Required argument

PRICE_SYMBOL
Required argument

nftsellbook

Returns the sell book for the given symbol

```
hiveengine nftsellbook [OPTIONS] SYMBOL
```

Options

-a, --account <account>
Buy with this account (defaults to "default_account")

-g, --grouping <grouping>
Can be set to a grouping parameter, or to parameter.value

-v, --value <value>
Set property value, can be used when grouping is set to a property parameter

-s, --price-symbol <price_symbol>
Limit to this price symbol

-n, --nft-id <nft_id>
Limit to this nft id

-c, --cheapest-only
Show only the cheapest open sell for each type

-m, --min-hive <min_hive>
Show only NFT which have a higher price

-l, --limit <limit>
Limit to shown entries

-i, --interactive
Show only the cheapest open sell for each type

Arguments

SYMBOL
Required argument

nfttrades

Returns the trades history

```
hiveengine nfttrades [OPTIONS] [SYMBOL]
```

Options

-a, --account <account>
Buy with this account (defaults to “default_account”)

Arguments

SYMBOL
Optional argument

richlist

Shows the richlist of a token

```
hiveengine richlist [OPTIONS] SYMBOL
```

Options

-t, --top <top>
Show only the top n accounts

Arguments

SYMBOL
Required argument

sell

Put a sell-order for a token to the hive-engine market

```
hiveengine sell [OPTIONS] [AMOUNT] [TOKEN] [PRICE]
```

Options

-a, --account <account>
Buy with this account (defaults to “default_account”)

Arguments

AMOUNT
Optional argument

TOKEN
Optional argument

PRICE
Optional argument

sellbook

Returns the sell book for the given token

```
hiveengine sellbook [OPTIONS] [TOKEN]
```

Options

-a, --account <account>
Buy with this account (defaults to "default_account")

Arguments

TOKEN
Optional argument

stake

stake a token / all tokens

```
hiveengine stake [OPTIONS] [AMOUNT] [TOKEN]
```

Options

-a, --account <account>
Stake token from this account

-r, --receiver <receiver>
Stake to this account (default is sender account)

Arguments

AMOUNT
Optional argument

TOKEN
Optional argument

tokenlist

Show list of all tokens

```
hiveengine tokenlist [OPTIONS]
```

transfer

Transfer a token

```
hiveengine transfer [OPTIONS] TO [AMOUNT] [TOKEN] [MEMO]
```

Options

-m, --memos <memos>
Can be used when all tokens should be send

-a, --account <account>
Transfer from this account

Arguments

TO
Required argument

AMOUNT
Optional argument

TOKEN
Optional argument

MEMO
Optional argument

unstake

unstake a token / all tokens

```
hiveengine unstake [OPTIONS] [AMOUNT] [TOKEN]
```

Options

-a, --account <account>
Transfer from this account

Arguments

AMOUNT
Optional argument

TOKEN
Optional argument

withdraw

Widthdraw SWAP.HIVE to account as HIVE.

```
hiveengine withdraw [OPTIONS] AMOUNT
```

Options

-a, --account <account>
withdraw from this account

Arguments

AMOUNT
Required argument

1.4 Tutorials

1.5 Modules

1.5.1 hiveengine Modules

hiveengine.api

class hiveengine.api.**Api**(*url=None*, *rpcurl=None*, *user=None*, *password=None*, ***kwargs*)
Bases: object

Access the hive-engine API

find(*contract_name*, *table_name*, *query={}*, *limit=1000*, *offset=0*, *indexes=[]*)
Get an array of objects that match the query from the table of the specified contract

find_all(*contract_name*, *table_name*, *query={}*)
Get an array of objects that match the query from the table of the specified contract

find_one(*contract_name*, *table_name*, *query={}*)
Get the object that matches the query from the table of the specified contract

get_block_info(*blocknumber*)
get the block with the specified block number of the sidechain

get_contract(*contract_name*)
Get the contract specified from the database

get_history(*account*, *symbol*, *limit=1000*, *offset=0*)
"Get the transaction history for an account and a token

get_latest_block_info()
get the latest block of the sidechain

get_status()
gets the status of the sidechain

get_transaction_info(*txid*)
Retrieve the specified transaction info of the sidechain

hiveengine.collection

class hiveengine.collection.**Collection**(*account*, *api=None*, *blockchain_instance=None*, *steem_instance=None*)

Bases: dict

Access the hive-engine NFT collection

Parameters

- **account** (*str*) – Name of the account
- **blockchain_instance** (*Hive*) – Hive instance

Wallet example:

```
from hiveengine.collection import Collection
collection = Collection("test")
print(collection)
```

burn(*nfts*)
Burn a token

Parameters **nfts** (*list*) – Amount to transfer

Transfer example:

```
from hiveengine.collection import Collection
from beem import Hive
active_wif = "5xxxx"
hive = Hive(keys=[active_wif])
collection = Collection("test", blockchain_instance=hive)
nfts = [{"symbol": "STAR", "ids": ["100"]}]
collection.burn(nfts)
```

change_account(*account*)
Changes the wallet account

delegate(*to*, *nfts*, *from_type='user'*, *to_type='user'*)
Delegate a token to another account.

Parameters

- **to** (*str*) – Recipient
- **nfts** (*list*) – Amount to transfer
- **from_type** (*str*) – (optional) user / contract
- **to_type** (*str*) – (optional) user / contract

Transfer example:

```
from hiveengine.collection import Collection
from beem import Hive
active_wif = "5xxxx"
hive = Hive(keys=[active_wif])
collection = Collection("test", blockchain_instance=hive)
```

(continues on next page)

(continued from previous page)

```
nfts = [{"symbol": "STAR", "ids": ["100"]}]
collection.delegate("test2", nfts)
```

get_collection()
Returns all token within the wallet as list

get_nft(*nft_id*, *symbol*)
Returns a token from the wallet. Is None when not available.

refresh()

set_id(*ssc_id*)
Sets the ssc id (default is ssc-mainnet-hive)

transfer(*to*, *nfts*, *from_type='user'*, *to_type='user'*)
Transfer a token to another account.

Parameters

- **to** (*str*) – Recipient
- **nfts** (*list*) – Amount to transfer
- **from_type** (*str*) – (optional) user / contract
- **to_type** (*str*) – (optional) user / contract

Transfer example:

```
from hiveengine.collection import Collection
from beem import Hive
active_wif = "5xxxx"
hive = Hive(keys=[active_wif])
collection = Collection("test", blockchain_instance=hive)
nfts = [{"symbol": "STAR", "ids": ["100"]}]
collection.transfer("test2", nfts)
```

undelegate(*to*, *nfts*, *from_type='user'*)
Undelegate a token to another account.

Parameters

- **to** (*str*) – Recipient
- **nfts** (*list*) – Amount to transfer
- **from_type** (*str*) – (optional) user / contract

Transfer example:

```
from hiveengine.collection import Collection
from beem import Hive
active_wif = "5xxxx"
hive = Hive(keys=[active_wif])
collection = Collection("test", blockchain_instance=hive)
nfts = [{"symbol": "STAR", "ids": ["100"]}]
collection.undelegate("test2", nfts)
```

hiveengine.exceptions

exception hiveengine.exceptions.**InsufficientTokenAmount**
Bases: Exception

Not suffienct amount for transfer in the wallet

exception hiveengine.exceptions.**InvalidTokenAmount**
Bases: Exception

Invalid token amount (not fitting precision or max supply)

exception hiveengine.exceptions.**MaxSupplyReached**
Bases: Exception

Only the token issuer is allowed to permit new tokens

exception hiveengine.exceptions.**NftDoesNotExists**
Bases: Exception

Nft does not (yet) exists

exception hiveengine.exceptions.**TokenDoesNotExists**
Bases: Exception

Token does not (yet) exists

exception hiveengine.exceptions.**TokenIssueNotPermitted**
Bases: Exception

Only the token issuer is allowed to permit new tokens

exception hiveengine.exceptions.**TokenNotInWallet**
Bases: Exception

The token is not in the account wallet

hiveengine.market

class hiveengine.market.**Market**(*api=None*, *blockchain_instance=None*, *steem_instance=None*)
Bases: list

Access the hive-engine market

Parameters **blockchain_instance** (*Hive*) – Hive instance

buy(*account*, *amount*, *symbol*, *price*)
Buy token for given price.

Parameters

- **account** (*str*) – account name
- **amount** (*float*) – Amount to withdraw
- **symbol** (*str*) – symbol
- **price** (*float*) – price

Buy example:

```
from hiveengine.market import Market
from beem import Steem
active_wif = "5xxxx"
stm = Steem(keys=[active_wif])
market = Market(blockchain_instance=stm)
market.buy("test", 1, "BEE", 0.95)
```

cancel (*account*, *order_type*, *order_id*)
Cancel buy/sell order.

Parameters

- **account** (*str*) – account name
- **order_type** (*str*) – sell or buy
- **order_id** (*int*) – order id

Cancel example:

```
from hiveengine.market import Market
from beem import Steem
active_wif = "5xxxx"
stm = Steem(keys=[active_wif])
market = Market(blockchain_instance=stm)
market.sell("test", "sell", 12)
```

deposit (*account*, *amount*)
Deposit HIVE to market in exchange for SWAP.HIVE.

Parameters

- **account** (*str*) – account name
- **amount** (*float*) – Amount to deposit

Deposit example:

```
from hiveengine.market import Market
from beem import Steem
active_wif = "5xxxx"
stm = Steem(keys=[active_wif])
market = Market(blockchain_instance=stm)
market.deposit("test", 1)
```

get_buy_book (*symbol*, *account=None*, *limit=100*, *offset=0*)
Returns the buy book for a given symbol. When account is set, the order book from the given account is shown.

get_metrics ()
Returns all token within the wallet as list

get_sell_book (*symbol*, *account=None*, *limit=100*, *offset=0*)
Returns the sell book for a given symbol. When account is set, the order book from the given account is shown.

get_trades_history (*symbol*, *account=None*, *limit=30*, *offset=0*)
Returns the trade history for a given symbol. When account is set, the trade history from the given account is shown.

refresh ()

sell(*account*, *amount*, *symbol*, *price*)
Sell token for given price.

Parameters

- **account** (*str*) – account name
- **amount** (*float*) – Amount to withdraw
- **symbol** (*str*) – symbol
- **price** (*float*) – price

Sell example:

```
from hiveengine.market import Market
from beem import Steem
active_wif = "5xxxx"
stm = Steem(keys=[active_wif])
market = Market(blockchain_instance=stm)
market.sell("test", 1, "BEE", 0.95)
```

set_id(*ssc_id*)
Sets the ssc id (default is ssc-mainnet-hive)

withdraw(*account*, *amount*)
Widthdraw SWAP.HIVE to account as HIVE.

Parameters

- **account** (*str*) – account name
- **amount** (*float*) – Amount to withdraw

Withdraw example:

```
from hiveengine.market import Market
from beem import Steem
active_wif = "5xxxx"
stm = Steem(keys=[active_wif])
market = Market(blockchain_instance=stm)
market.withdraw("test", 1)
```

hiveengine.nft

class hiveengine.nft.**Nft**(*symbol*, *api=None*, *blockchain_instance=None*)
Bases: dict

Access the hive-engine Nfts

add_authorized_issuing_accounts(*accounts*)
Adds Hive accounts to the list of accounts that are authorized to issue new tokens on behalf of the NFT owner.

param list accounts A list of hive accounts to add to the authorized list

example:

```
from hiveengine.nft import Nft
from beem import Hive
active_wif = "5xxxx"
```

(continues on next page)

(continued from previous page)

```
hive = Hive(keys=[active_wif])
nft = Nft("TESTNFT", blockchain_instance=hive)
nft.add_authorized_issuing_accounts(["satoshi","aggroed","cryptomancer
↪"])
```

add_authorized_issuing_contracts(*contracts*)

Adds smart contracts to the list of contracts that are authorized to issue new tokens on behalf of the NFT owner.

param list contracts A list of smart contracts t to add to the authorized list

example:

```
from hiveengine.nft import Nft
from beem import Hive
active_wif = "5xxxx"
hive = Hive(keys=[active_wif])
nft = Nft("TESTNFT", blockchain_instance=hive)
nft.add_authorized_issuing_contracts(["mycontract","anothercontract",
↪"mygamecontract"])
```

add_property(*name*, *prop_type*, *is_read_only=None*, *authorized_editing_accounts=None*, *authorized_editing_contracts=None*)

Adds a new data property schema to an existing NFT definition

Parameters

- **name** (*str*) – Name of the new property
- **prop_type** (*str*) – must be number, string or boolean
- **is_read_only** (*bool*) –
- **authorized_editing_accounts** (*list*) –
- **authorized_editing_contracts** (*list*) –

example:

```
from hiveengine.nft import Nft
from beem import Hive
active_wif = "5xxxx"
hive = Hive(keys=[active_wif])
nft = Nft("TESTNFT", blockchain_instance=hive)
nft.add_property("color", "string")
```

enable_delegation(*undelegation_cooldown*)

Enables the delegation feature for a NFT

Parameters **undelegation_cooldown** (*int*) – Cooldown in days

example:

```
from hiveengine.nft import Nft
from beem import Hive
active_wif = "5xxxx"
hive = Hive(keys=[active_wif])
nft = Nft("TESTNFT", blockchain_instance=hive)
nft.enable_delegation(30)
```

get_collection(*account*)
Get NFT collection

get_id(*_id*)
Get info about a token

get_info()
Returns information about the nft

get_open_interest(*query={}*, *limit=-1*, *offset=0*)
Returns open interests :param dict query: side, priceSymbol, grouping

get_property(*property_name*)
Returns all token properties

get_sell_book(*query={}*, *limit=-1*, *offset=0*)
Returns the sell book :param dict query: can be ownedBy, account, nftId, grouping, priceSymbol

get_trade_history(*query={}*, *limit=-1*, *offset=0*)
Returns market information :param dict query: can be priceSymbol, timestamp

issue(*to*, *fee_symbol*, *from_type=None*, *to_type=None*, *lock_tokens=None*, *lock_nfts=None*, *properties=None*, *authorized_account=None*)
Issues a new instance of an NFT to a Hive account or smart contract.

Parameters

- **to** (*str*) –
- **fee_symbol** (*str*) –
- **from_type** (*str*) –
- **to_type** (*str*) –
- **lock_tokens** (*dict*) –
- **lock_nfts** (*list*) –
- **properties** (*dict*) –
- **authorized_account** (*str*) – authorized hive account

example:

```
from hiveengine.nft import Nft
from beem import Hive
active_wif = "5xxxx"
hive = Hive(keys=[active_wif])
nft = Nft("TESTNFT", blockchain_instance=hive)
nft.issue("aggroed", "PAL")
```

issue_multiple(*instances*, *authorized_account=None*)
Issues multiple NFT instances at once.

Parameters

- **instances** (*list*) –
- **authorized_account** (*str*) – authorized hive account

example:

```
from hiveengine.nft import Nft
from beem import Hive
active_wif = "5xxxx"
hive = Hive(keys=[active_wif])
nft = Nft("TESTNFT", blockchain_instance=hive)
nft.issue_multiple([{"fromType": "contract", "symbol": "TSTNFT", "to": "marc",
↪ "feeSymbol": "PAL"}])
```

issuer

properties

refresh()

remove_authorized_issuing_accounts(*accounts*)

Removes Hive accounts from the list of accounts that are authorized to issue new tokens on behalf of the NFT owner.

param list accounts A list of hive accounts to remove from the authorized list

example:

```
from hiveengine.nft import Nft
from beem import Hive
active_wif = "5xxxx"
hive = Hive(keys=[active_wif])
nft = Nft("TESTNFT", blockchain_instance=hive)
nft.remove_authorized_issuing_accounts(["aggroed","cryptomancer"])
```

remove_authorized_issuing_contracts(*contracts*)

Remvoes smart contracts from the list of contracts that are authorized to issue new tokens on behalf of the NFT owner.

param list contracts A list of smart contracts to remove from the authorized list

example:

```
from hiveengine.nft import Nft
from beem import Hive
active_wif = "5xxxx"
hive = Hive(keys=[active_wif])
nft = Nft("TESTNFT", blockchain_instance=hive)
nft.remove_authorized_issuing_contracts(["mycontract","mygamecontract
↪"])
```

set_group_by(*properties*)

Can be used after calling the addProperty action to change the lists of authorized editing accounts & contracts for a given data property.

param list properties

example:

```
from hiveengine.nft import Nft
from beem import Hive
active_wif = "5xxxx"
hive = Hive(keys=[active_wif])
nft = Nft("TESTNFT", blockchain_instance=hive)
nft.set_group_by(["level", "isFood"])
```

set_properties(*nfts*, *from_type=None*, *authorized_account=None*)
Edits one or more data properties on one or more instances of an NFT.

Parameters

- **nfts** (*list*) –
- **from_type** (*str*) –
- **authorized_account** (*str*) – authorized hive account

example:

```
from hiveengine.nft import Nft
from beem import Hive
posting_wif = "5xxxx"
hive = Hive(keys=[posting_wif])
nft = Nft("TESTNFT", blockchain_instance=hive)
nft.set_properties([{ "id":"573", "properties": {"color": "red", "level": 2}}
↪])
```

set_property_permissions(*name*, *accounts=None*, *contracts=None*)
Can be used after calling the addProperty action to change the lists of authorized editing accounts & contracts for a given data property.

param str name Name of the new property

param list accounts

param list contracts

example:

```
from hiveengine.nft import Nft
from beem import Hive
active_wif = "5xxxx"
hive = Hive(keys=[active_wif])
nft = Nft("TESTNFT", blockchain_instance=hive)
nft.set_property_permissions("color", accounts=["cryptomancer","marc"])
```

transfer_ownership(*to*)
Transfers ownership of an NFT from the current owner to another Hive account.

Parameters **to** (*str*) – Hive accounts to become the new owner

example:

```
from hiveengine.nft import Nft
from beem import Hive
active_wif = "5xxxx"
hive = Hive(keys=[active_wif])
nft = Nft("TESTNFT", blockchain_instance=hive)
nft.transfer_ownership("aggroed")
```

update_metadata(*medadata*)
Updates the metadata of a token.

Parameters **medadata** (*dict*) – new medadata

example:

```
from hiveengine.nft import Nft
from beem import Hive
active_wif = "5xxxx"
hive = Hive(keys=[active_wif])
nft = Nft("STAR", blockchain_instance=hive)
metadata = {"url": "https://mycoolnft.com",
            "icon": "https://mycoolnft.com/token.jpg",
            "desc": "This NFT will rock your world! It has features x, y, and␣
↪z. So cool!"}
nft.update_metadata(metadata)
```

update_name(*name*)
Updates the user friendly name of an NFT.

Parameters **name** (*str*) – new name

example:

```
from hiveengine.nft import Nft
from beem import Hive
posting_wif = "5xxxx"
hive = Hive(keys=[posting_wif])
nft = Nft("STAR", blockchain_instance=hive)
nft.update_name("My Awesome NFT")
```

update_org_name(*org_name*)
Updates the name of the company/organization that manages an NFT.

Parameters **org_name** (*str*) – new org_name

example:

```
from hiveengine.nft import Nft
from beem import Hive
posting_wif = "5xxxx"
hive = Hive(keys=[posting_wif])
nft = Nft("STAR", blockchain_instance=hive)
nft.update_org_name("Nifty Company Inc")
```

update_product_name(*product_name*)
Updates the name of the company/organization that manages an NFT.

Parameters **org_name** (*str*) – new org_name

example:

```
from hiveengine.nft import Nft
from beem import Hive
posting_wif = "5xxxx"
hive = Hive(keys=[posting_wif])
nft = Nft("STAR", blockchain_instance=hive)
nft.update_product_name("Acme Exploding NFTs")
```

update_property_definition(*name*, *new_name=None*, *prop_type=None*, *is_read_only=None*)
Updates the schema of a data property. This action can only be called if no tokens for this NFT have been issued yet.

Parameters

- **name** (*str*) – Name of the new property

- **name** –
- **new_name** (*str*) –
- **prop_type** (*str*) –
- **is_read_only** (*bool*) –

example:

```
from hiveengine.nft import Nft
from beem import Hive
active_wif = "5xxxx"
hive = Hive(keys=[active_wif])
nft = Nft("TESTNFT", blockchain_instance=hive)
nft.update_property_definition("color", new_name="Color")
```

update_url(*url*)

Updates the NFT project website

Parameters **url** (*str*) – new url

example:

```
from hiveengine.nft import Nft
from beem import Hive
active_wif = "5xxxx"
hive = Hive(keys=[active_wif])
nft = Nft("STAR", blockchain_instance=hive)
nft.update_url("https://new_url.com")
```

hiveengine.rpc

graphennewsrpc.

class hiveengine.rpc.**RPC**(*url=None*, *user=None*, *password=None*, ***kwargs*)

Bases: object

This class allows to call API methods synchronously, without callbacks.

It logs warnings and errors.

Usage:

```
from hiveengine.rpc import RPC
rpc = RPC()
print(rpc.getLatestBlockInfo(endpoint="blockchain"))
```

get_request_id()

Get request id.

request_send(*endpoint*, *payload*)

rpcexec(*endpoint*, *payload*)

Execute a call by sending the payload.

Parameters **payload** (*json*) – Payload data

Raises

- **ValueError** – if the server does not respond in proper JSON format
- ***RPCError*** – if the server returns an error

version_string_to_int(*network_version*)

exception hiveengine.rpc.**RPCError**
Bases: Exception

RPCError Exception.

exception hiveengine.rpc.**RPCErrorDoRetry**
Bases: Exception

RPCErrorDoRetry Exception.

class hiveengine.rpc.**SessionInstance**
Bases: object

Singelton for the Session Instance

instance = None

exception hiveengine.rpc.**UnauthorizedError**
Bases: Exception

UnauthorizedError Exception.

hiveengine.rpc.**get_endpoint_name**(**args*, ***kwargs*)

hiveengine.rpc.**set_session_instance**(*instance*)
Set session instance

hiveengine.rpc.**shared_session_instance**()
Get session instance

hiveengine.tokenobject

class hiveengine.tokenobject.**Token**(*symbol*, *api=None*)
Bases: dict

hive-engine token dict

Parameters **token** (*str*) – Name of the token

get_buy_book(*limit=100*, *offset=0*)
Returns the buy book

get_holder(*limit=1000*, *offset=0*)
Returns all token holders

get_info()
Returns information about the token

get_market_info()
Returns market information

get_sell_book(*limit=100*, *offset=0*)
Returns the sell book

quantize(*amount*)
Round down a amount using the token precision and returns a Decimal object

refresh()

hiveengine.tokens

class hiveengine.tokens.**Tokens**(*api=None*, ***kwargs*)
Bases: list

Access the steem-engine tokens

get_token(*symbol*)
Returns Token from given token symbol. Is None when token does not exists.

get_token_list()
Returns all available token as list

refresh()

hiveengine.wallet

class hiveengine.wallet.**Wallet**(*account*, *api=None*, *blockchain_instance=None*, *steem_instance=None*)
Bases: list

Access the hive-engine wallet

Parameters

- **account** (*str*) – Name of the account
- **blockchain_instance** (*Hive*) – Hive instance

Wallet example:

```
from hiveengine.wallet import Wallet
wallet = Wallet("test")
print(wallet)
```

cancel_unstake(*trx_id*)
Cancel unstaking a token.

Parameters **trx_id** (*str*) – transaction id in which the tokan was unstaked

Cancel unstake example:

```
from hiveengine.wallet import Wallet
from beem import Steem
active_wif = "5xxxx"
stm = Steem(keys=[active_wif])
wallet = Wallet("test", blockchain_instance=stm)
wallet.stake("cf39ecb8b846f1efffb8db526fada21a5fcf41c3")
```

change_account(*account*)
Changes the wallet account

get_balances()
Returns all token within the wallet as list

get_buy_book(*symbol=None*, *limit=100*, *offset=0*)
Returns the buy book for the wallet account. When symbol is set, the order book from the given token is shown.

get_history(*symbol*, *limit=1000*, *offset=0*)
Returns the transfer history of a token

get_sell_book (*symbol=None*, *limit=100*, *offset=0*)
Returns the sell book for the wallet account. When symbol is set, the order book from the given token is shown.

get_token (*symbol*)
Returns a token from the wallet. Is None when not available.

issue (*to*, *amount*, *symbol*)
Issues a specific token amount.

Parameters

- **to** (*str*) – Recipient
- **amount** (*float*) – Amount to issue
- **symbol** (*str*) – Token to issue

Issue example:

```
from hiveengine.wallet import Wallet
from beem import Steem
active_wif = "5xxxx"
stm = Steem(keys=[active_wif])
wallet = Wallet("test", blockchain_instance=stm)
wallet.issue(1, "my_token")
```

refresh ()

set_id (*ssc_id*)
Sets the ssc id (default is ssc-mainnet-hive)

stake (*amount*, *symbol*, *receiver=None*)
Stake a token.

Parameters

- **amount** (*float*) – Amount to stake
- **symbol** (*str*) – Token to stake

Stake example:

```
from hiveengine.wallet import Wallet
from beem import Steem
active_wif = "5xxxx"
stm = Steem(keys=[active_wif])
wallet = Wallet("test", blockchain_instance=stm)
wallet.stake(1, "BEE")
```

transfer (*to*, *amount*, *symbol*, *memo=''*)
Transfer a token to another account.

Parameters

- **to** (*str*) – Recipient
- **amount** (*float*) – Amount to transfer
- **symbol** (*str*) – Token to transfer
- **memo** (*str*) – (optional) Memo

Transfer example:

```
from hiveengine.wallet import Wallet
from beem import Steem
active_wif = "5xxxx"
stm = Steem(keys=[active_wif])
wallet = Wallet("test", blockchain_instance=stm)
wallet.transfer("test1", 1, "BEE", "test")
```

unstake (*amount*, *symbol*)
Unstake a token.

Parameters

- **amount** (*float*) – Amount to unstake
- **symbol** (*str*) – Token to unstake

Unstake example:

```
from hiveengine.wallet import Wallet
from beem import Steem
active_wif = "5xxxx"
stm = Steem(keys=[active_wif])
wallet = Wallet("test", blockchain_instance=stm)
wallet.unstake(1, "BEE")
```

1.6 Contributing to hiveengine

We welcome your contributions to our project.

1.6.1 Repository

The repository of beem is currently located at:

https://github.com/holgern/hiveengine

1.6.2 Flow

This project makes heavy use of git flow. If you are not familiar with it, then the most important thing for your to understand is that:

pull requests need to be made against the develop branch

1.6.3 How to Contribute

0. Familiarize yourself with contributing on github
1. Fork or branch from the master.
2. Create commits following the commit style
3. Start a pull request to the master branch
4. Wait for a @holger80 or another member to review

1.6.4 Issues

Feel free to submit issues and enhancement requests.

1.6.5 Contributing

Please refer to each project's style guidelines and guidelines for submitting patches and additions. In general, we follow the "fork-and-pull" Git workflow.

1. **Fork** the repo on GitHub
2. **Clone** the project to your own machine
3. **Commit** changes to your own branch
4. **Push** your work back up to your fork
5. Submit a **Pull request** so that we can review your changes

Note: Be sure to merge the latest from "upstream" before making a pull request!

1.6.6 Copyright and Licensing

This library is open sources under the MIT license. We require your to release your code under that license as well.

1.7 Support and Questions

Help and discussion channel for hiveengine can be found here:

- https://discord.gg/4HM592V

1.8 Indices and Tables

- genindex
- modindex

CHAPTER 2

Indices and tables

- genindex
- modindex
- search

Python Module Index

h

Index

Symbols

A

B

C

D

E

F

G

H

I

M

N

O

P

Q

R

S

T