Privex Python Steemengine Documentation

Privex Inc., Chris (Someguy123)

Aug 06, 2022

MAIN:

Welcome to the documentation for Privex's Steem-Engine library - a python package built for interacting with the *Steem-Engine* smart contract network. It includes support for sending/issuing tokens, obtaining information about tokens, checking balances, and loading the transaction history for any account.

This documentation is automatically kept up to date by ReadTheDocs, as it is automatically re-built each time a new commit is pushed to the Github Project

CHAPTER

ONE

CONTENTS

1.1 Installation

1.1.1 Download and install from PyPi using pip (recommended)

```
pip3 install privex-steemengine
```

1.1.2 (Alternative) Manual install from Git

Option 1 - Use pip to install straight from Github

```
pip3 install git+https://github.com/Privex/python-steemengine
```

Option 2 - Clone and install manually

```
# Clone the repository from Github
git clone https://github.com/Privex/python-steemengine
cd python-steemengine

# RECOMMENDED MANUAL INSTALL METHOD
# Use pip to install the source code
pip3 install .

# ALTERNATIVE MANUAL INSTALL METHOD
# If you don't have pip, or have issues with installing using it, then you can use␣
↪setuptools instead.
python3 setup.py install
```

privex.steemengine.SteemEngineToken.SteemEngineToken([...])	SteemEngineToken - a wrapper class around `privex.jsonrpc.SteemEngineRPC`, with support for signing transactions, including issuing/sending tokens.
privex.steemengine.SteemEngineHistory.SteemEngineHistory([...])	Provides simple methods for querying a SteemEngine history node
privex.steemengine.exceptions	Copyright.
privex.steemengine.objects	This file contains various classes used to represent different dictionaries returned from the Steem/HiveEngine APIs, making the data easier to handle.
tests	Unit tests for Privex's SteemEngine library.

1.2 privex.steemengine.SteemEngineToken.SteemEngineToken

class SteemEngineToken(*network_account='ssc-mainnet1'*, *history_conf: Optional[dict] = None*, *network='steem'*, ***rpc_args*)

SteemEngineToken - a wrapper class around `privex.jsonrpc.SteemEngineRPC`, with support for signing transactions, including issuing/sending tokens.

Copyright:

```
+=====================================================+
|                 © 2019 Privex Inc.                  |
|               https://www.privex.io                 |
+=====================================================+
|                                                     |
|        Python Steem Engine Library                  |
|        License: X11/MIT                             |
|                                                     |
|        Core Developer(s):                           |
|                                                     |
|          (+)  Chris (@someguy123) [Privex]          |
|                                                     |
+=====================================================+
```

Basic usage:

```
>>> from privex.steemengine import SteemEngineToken
>>> s = SteemEngineToken()
>>> # Send 10 ENG to @privex from @someguy123 with the memo 'hello memo'
>>> s.send_token('ENG', 'someguy123', 'privex', Decimal('10'), 'hello memo')
```

__init__(*network_account='ssc-mainnet1'*, *history_conf: Optional[dict] = None*, *network='steem'*, ***rpc_args*)

Initialises the class with various configuration options. All parameters are optional.

Pass a dict in `history_conf` to override the SteemEngine history node used Pass extra kwargs such as `hostname='api.example.com'` ot override the SteemEngine RPC node used.

Parameters

- **network_account** – The Steem account that operates the SteemEngine network, e.g. ssc-mainnet1 for the Steem Smart Contracts Main Network
- **history_conf** – A dictionary containing kwargs to pass to *SteemEngineHistory* constructor
- **rpc_args** – Any additional kwargs will be passed to the `privex.jsonrpc.SteemEngineRPC` constructor
- **nodes** (*str*|*list*) – If the `nodes` kwarg is specified, e.g. `nodes=['https://api.something.com']`, then Beem will be configured during `__init__` to use those specific RPC nodes.

Methods

`__init__`([network_account, history_conf, ...])	Initialises the class with various configuration options.
`account_exists`(user)	Helper function to verify if a given user exists on Steem.
`custom_beem`([node, network])	Generates a new Beem `Steem` / `Hive` instance
`find_fulfilled`(txid[, limit, offset, indexes])	
`find_fulfilled_buys`(txid[, limit, offset, ...])	
`find_fulfilled_sells`(txid[, limit, offset, ...])	
`find_steem_tx`(tx_data[, last_blocks])	Used internally to get the transaction ID after a Steem transaction has been broadcasted.
`get_balances`(user)	Get all token balances for a user.
`get_orderbook`(symbol[, direction, user, ...])	Get a list of open Steem/Hive Engine orders for a given symbol, by default will display `'buy'` orders unless you set `direction` to `'sell'`
`get_ticker`(symbol, **query)	Retrieve a ticker for a single symbol as a `SETicker` object.
`get_tickers`([limit, offset, indexes])	Retrieve a list of market tickers from Hive/SteemEngine as a list of `SETicker` objects.
`get_token`(symbol)	Get the token object for an individual token.
`get_token_balance`(user, symbol)	Find a specific token balance of a user.
`get_transaction_info`(txid)	
`issue_token`(symbol, to, amount[, find_tx])	Issues a specified amount `amount` of `symbol` to the Steem account `to`.
`list_tokens`([limit, offset])	Returns a list of all tokens as `Token` objects.
`list_transactions`(user[, symbol, limit, offset])	Get the Steem Engine transaction history for a given account
`order_history`(symbol[, limit, offset, indexes])	Get a list of recent Steem Engine orders for a given symbol.
`place_order`(user, action, symbol, quantity, ...)	Place an order on the Steem/Hive Engine market
`query_order_history`([limit, offset, indexes])	Used internally by methods such as `order_history()` and `find_fulfilled_sells()` etc.
`send_token`(symbol, from_acc, to_acc, amount)	Sends a given `amount` of `symbol` from `from_acc` to `to_acc` with the memo `memo`.
`set_beem`([nodes, network])	Create and override the Beem instance used by this instance.
`trade_history`(symbol[, limit, offset, indexes])	Get a list of recent Steem Engine orders for a given symbol.
`verify_network`([network, inst])	Check that the current RPC node is on the correct blockchain network.

Attributes

CACHE	Global cache switch.
CACHE_BLACKLIST	A list of fully qualified module paths to functions/methods which always bypass `se_cache()`
CACHE_BLACKLIST_FUNCS	A list of plain function names which always bypass `se_cache()`
CACHE_BLACKLIST_MODS	A list of fully qualified module names which always bypass `se_cache()`
DEFAULT_BLOCKCHAIN_URL	
default_nodes	This is a class-level attribute (shared by all instances, instead of specific to one instance), which contains a list of RPC nodes to be used by default for each network.
native_token	Returns the `Token` object for the native coin `native_coin`
steem	When a method calls `self.steem`, this property will first try to return `_steem` if it was previously called.
use_shared_instances	This is a class-level attribute that controls whether instances of *SteemEngineToken* should use Beem's shared instances.
network	
native_coin	

1.3 privex.steemengine.SteemEngineHistory.SteemEngineHistory

class SteemEngineHistory(*hostname='api.steem-engine.net'*, *port: int = 443*, *username=None*, *password=None*, *ssl=True*, *timeout=120*, *url: str = 'accounts/history'*)

Provides simple methods for querying a SteemEngine history node

Copyright:

```
+=====================================================+
|                 © 2019 Privex Inc.                  |
|               https://www.privex.io                 |
+=====================================================+
|                                                     |
|        Python Steem Engine Library                  |
|        License: X11/MIT                             |
|                                                     |
|        Core Developer(s):                           |
|                                                     |
|          (+)  Chris (@someguy123) [Privex]          |
|                                                     |
+=====================================================+
```

Basic Usage:

```
>>> h = SteemEngineHistory()
>>> for tx in h.get_history('someguy123'):
>>>     print(tx['timestamp'], tx['symbol'], tx['quantity'], tx['memo'])
```

__init__(*hostname='api.steem-engine.net'*, *port: int = 443*, *username=None*, *password=None*, *ssl=True*, *timeout=120*, *url: str = 'accounts/history'*)

Configure the remote SteemEngine history server settings. All parameters are optional.

Parameters

- **hostname** – The hostname or IP address of the history server
- **port** – The HTTP port to connect to
- **username** – If the RPC server needs a username, specify it here
- **password** – If the RPC server needs a password, specify it here (username must also be set)
- **ssl** – If set to True, will use https for requests. Default is True - use SSL
- **timeout** – If the server stops sending us data for this many seconds, abort and throw an exception
- **url** – The URL to query, e.g. accounts/history (starting /'s will automatically be removed)

Methods

`__init__`([hostname, port, username, ...])	Configure the remote SteemEngine history server settings.
`get_history`(account[, symbol, limit, ...])	Get the Steem Engine transaction history for a given account :param account: Account name to filter by :param symbol: Symbol to filter by, e.g.

Attributes

`DEF_HOST`
`DEF_URL`
`req`
`url`

1.4 privex.steemengine.exceptions

Copyright:

```
+=================================================+
|                 © 2019 Privex Inc.              |
|               https://www.privex.io             |
+=================================================+
|                                                 |
|        Python Steem Engine library              |
|        License: X11/MIT                         |
|                                                 |
|        Core Developer(s):                       |
|                                                 |
|          (+)  Chris (@someguy123) [Privex]      |
|                                                 |
+=================================================+
```

Python SteemEngine - A small library for querying and interacting with the SteemEngine network (https://steem-engine.com) Copyright (c) 2019 Privex Inc. (https://www.privex.io)

Permission is hereby granted, free of charge, to any person obtaining a copy of this software and associated documentation files (the "Software"), to deal in the Software without restriction, including without limitation the rights to use, copy, modify, merge, publish, distribute, sublicense, and/or sell copies of the Software, and to permit persons to whom the Software is furnished to do so, subject to the following conditions:

The above copyright notice and this permission notice shall be included in all copies or substantial portions of the Software.

THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE SOFTWARE.

Except as contained in this notice, the name(s) of the above copyright holders shall not be used in advertising or otherwise to promote the sale, use or other dealings in this Software without prior written authorization.

Exceptions

`AccountNotFound`	The Steem account requested doesn't exist
`NoResults`	The server returned an empty response such as `None` ...
`NoSteemEngineInstance`	Raised when `_seng_instance` on a `SteemEngineInstanceInject` based object is `None`
`NotEnoughBalance`	Not enough token/steem/sbd balance for this operation
`SteemEngineException`	Base exception for all `privex.steemengine` exceptions
`TokenNotFound`	The token requested doesn't exist
`WrongNetwork`	The current RPC is on the wrong network, high risk of broadcasting or receiving data to/from the wrong blockchain network

1.5 privex.steemengine.objects

This file contains various classes used to represent different dictionaries returned from the Steem/HiveEngine APIs, making the data easier to handle:

```
* IDE attribute assistance (aka IntelliSense),
* converting certain attributes into more sensible types, e.g. from :class:`.str` /␣
↪:class:`.float` into :class:`.Decimal`
* new properties / methods to automate certain data queries and ease conversion to/from␣
↪different types.
```

Functions

conv_dec(number)	Convert an object into a Decimal, but only if it isn't already a Decimal

Classes

ObjBase([raw_data])	A base class to be extended by data storage classes, allowing their attributes to be accessed as if the class was a dict/list.
SEBalance(account, symbol, balance, float, ...)	Represents an account token balance on SteemEngine
SEContractTransfer(sender, to, symbol, ...)	Represents the data for a transferToContract / transferFromContract SETransactionLogEvent
SEOrder(symbol, quantity, float, str], ...)	Represents an open order on the SE market.
SEPlacedOrder(symbol, quantity, price, ...)	
SETicker(symbol, volume, lastPrice, ...)	
SETrade(symbol, quantity, float, str], ...)	Represents a past trade on the SE market.
SETransaction(block, txid, symbol, sender, ...)	Represents a standard transaction from account history on SteemEngine
SETransactionInfo(blockNumber, ...)	Represents transaction data from /rpc/blockchain JSON-RPC method getTransactionInfo
SETransactionLogEvent(contract, event, data, ...)	Represents events contained within the list SETransactionInfo .logs['events']
SteemEngineInstanceInject()	
Token(symbol, name, issuer, metadata, str, ...)	Represents a token's information on SteemEngine
TokenMetadata(url, icon, desc, raw_data, ...)	Represents the metadata field on a token object on SteemEngine

1.6 tests

Unit tests for Privex's SteemEngine library.

To run them, use pytest

```
# With pytest
pip3 install pytest pytest-cov pytest-asyncio coverage codecov
pytest
# Verbose mode
pytest -v
```

CHAPTER

TWO

INDICES AND TABLES

- genindex
- modindex
- search

PYTHON MODULE INDEX

p

t

INDEX

Symbols

M

P

S

T