TronAPI for Python Documentation

Release latest

Nov 11, 2018

Contents

TronAPI is a python library for interacting with Tron Protocol.

CHAPTER 1

Contents

1.1 Quickstart

Note: All code starting with a `$` is meant to run on your terminal. All code starting with a >>> is meant to run in a python interpreter, like ipython.

1.1.1 Installation

TronAPI can be installed (preferably in a virtualenv) using `pip` as follows:

```
$ pip install tronapi
```

Note: If you run into problems during installation, you might have a broken environment. See the troubleshooting guide to setup_environment.

Installation from source can be done from the root of the project with the following command.

```
$ pip install .
```

1.1.2 Using TronAPI

To use the tron library you will need to initialize the `tronapi` class.

```
>>> from tronapi import Tron
>>> full_node = HttpProvider('https://api.trongrid.io')
>>> solidity_node = HttpProvider('https://api.trongrid.io')
>>> event_server = 'https://api.trongrid.io'
>>>
>>> tron = Tron(full_node, solidity_node, event_server)
>>> tron.default_block = 'latest'
```

1.1.3 Getting Blockchain Info

It's time to start using TronAPI for Python! Try getting all the information about the latest block.

```
>>> tron.get_block('latest')
>>> {
"blockID": "00000000003a5bbda4aea15cb5d99230674463e9d5f2c0c647316839b25fd5b9",
"block_header": {
    "raw_data": {
        "number": 3824573,
        "txTrieRoot":
↪"31ee3e2ed28f843bf1d53495beece2f5b9c76480772f0106e17156fb0066c3a2",
        "witness_address": "41f70386347e689e6308e4172ed7319c49c0f66e0b",
        "parentHash":
↪"00000000003a5bbc1e78e3144ad52f01a27b8f7acceb98d3ca09c1abea5cd32a",
        "version": 3,
        "timestamp": 1541425827000
    },
    "witness_signature":
↪"fddc729f55c0ecc6f9cf4ab17cf818ddc0e85d2c21382ed6b1430adb1dcd13006c24ae0e08f16d29362452ec8869d29a28
↪"
},
"transactions": [

]
```

}

1.2 Overview

The common entrypoint for interacting with the Tron library is the `Tron` object. The tron object provides APIs for interacting with the tron blockchain, typically by connecting to a HTTP server.

1.2.1 Providers

Providers are how tron connects to the blockchain. The TronAPI library comes with a the following built-in providers that should be suitable for most normal use cases.

- `HttpProvider` for connecting to http and https based servers.

The `HttpProvider` takes the full URI where the server can be found. For local development this would be something like `http://localhost:8090`.

```
>>> from tronapi import HttpProvider, Tron

# Note that you should create only one HttpProvider per
# process, as it recycles underlying TCP/IP network connections between
# your process and Tron node

>>> full_node = HttpProvider('http://localhost:8090')
>>> solidity_node = HttpProvider('http://localhost:8090')
>>> event_server = HttpProvider('http://localhost:8090')

>>> tron = Tron(full_node, solidity_node, event_server)
```

1.2.2 Base API

The `Tron` class exposes the following convenience APIs.

Type Conversions

`Tron.`**`toHex`**(*primitive=None*, *hexstr=None*, *text=None*)

Takes a variety of inputs and returns it in its hexadecimal representation.

```
>>> Tron.toHex(0)
'0x0'
>>> Tron.toHex(1)
'0x1'
>>> Tron.toHex(0x0)
'0x0'
>>> Tron.toHex(0x000F)
'0xf'
>>> Tron.toHex(b'')
'0x'
>>> Tron.toHex(b'\x00\x0F')
'0x000f'
>>> Tron.toHex(False)
'0x0'
>>> Tron.toHex(True)
'0x1'
>>> Tron.toHex(hexstr='0x000F')
'0x000f'
>>> Tron.toHex(hexstr='000F')
'0x000f'
```

(continues on next page)

(continued from previous page)

```
>>> Tron.toHex(text='')
'0x'
>>> Tron.toHex(text='cowmö')
'0x636f776dc3b6'
```

`Tron.`**`toText`**(*primitive=None*, *hexstr=None*, *text=None*)

Takes a variety of inputs and returns its string equivalent. Text gets decoded as UTF-8.

```
>>> Tron.toText(0x636f776dc3b6)
'cowmö'
>>> Tron.toText(b'cowm\xc3\xb6')
'cowmö'
>>> Tron.toText(hexstr='0x636f776dc3b6')
'cowmö'
>>> Tron.toText(hexstr='636f776dc3b6')
'cowmö'
>>> Tron.toText(text='cowmö')
'cowmö'
```

`Tron.`**`toBytes`**(*primitive=None*, *hexstr=None*, *text=None*)

Takes a variety of inputs and returns its bytes equivalent. Text gets encoded as UTF-8.

```
>>> Tron.toBytes(0)
b'\x00'
>>> Tron.toBytes(0x000F)
b'\x0f'
>>> Tron.toBytes(b'')
b''
>>> Tron.toBytes(b'\x00\x0F')
b'\x00\x0f'
>>> Tron.toBytes(False)
b'\x00'
>>> Tron.toBytes(True)
b'\x01'
>>> Tron.toBytes(hexstr='0x000F')
b'\x00\x0f'
>>> Tron.toBytes(hexstr='000F')
b'\x00\x0f'
>>> Tron.toBytes(text='')
b''
>>> Tron.toBytes(text='cowmö')
b'cowm\xc3\xb6'
```

`Tron.`**`toInt`**(*primitive=None*, *hexstr=None*, *text=None*)

Takes a variety of inputs and returns its integer equivalent.

```
>>> Tron.toInt(0)
0
>>> Tron.toInt(0x000F)
15
>>> Tron.toInt(b'\x00\x0F')
15
>>> Tron.toInt(False)
0
>>> Tron.toInt(True)
1
```

(continues on next page)

(continued from previous page)

```
>>> Tron.toInt(hexstr='0x000F')
15
>>> Tron.toInt(hexstr='000F')
15
```

Currency Conversions

`Tron.`**`toSun`**(*value*)

Returns the value in the denomination specified by the `currency` argument converted to sun.

```
>>> tron.toSun(1)
1000000
```

`Tron.`**`fromSun`**(*value*)

Returns the value in wei converted to the given currency. The value is returned as a `Decimal` to ensure precision down to the wei.

```
>>> tron.fromSun(1000000)
Decimal('1')
```

Addresses

`Tron.`**`isAddress`**(*value*)

Returns `True` if the value is one of the recognized address formats.

```
>>> tron.isAddress('TRWBqiqoFZysoAeyR1J35ibuyc8EvhUAoY')
True
```

Cryptographic Hashing

`classmethod` `Tron.`**`sha3`**(*primitive=None*, *hexstr=None*, *text=None*)

Returns the Keccak SHA256 of the given value. Text is encoded to UTF-8 before computing the hash, just like Solidity. Any of the following are valid and equivalent:

```
>>> Tron.sha3(0x747874)
>>> Tron.sha3(b'\x74\x78\x74')
>>> Tron.sha3(hexstr='0x747874')
>>> Tron.sha3(hexstr='747874')
>>> Tron.sha3(text='txt')
HexBytes('0xd7278090a36507640ea6b7a0034b69b0d240766fa3f98e3722be93c613b29d2e')
```

1.3 Migrating your code from v1 to v2

1.3.1 Changes to base API convenience methods

Tron.toDecimal()

In v4 `Tron.toDecimal()` is renamed: *`toInt()`* for improved clarity. It does not return a `decimal.Decimal`, it returns an `int`.

Removed Methods

- `Tron.toUtf8` was removed for *`toText()`*.
- `Tron.fromUtf8` was removed for *`toHex()`*.
- `Tron.toAscii` was removed for *`toBytes()`*.
- `Tron.fromAscii` was removed for *`toHex()`*.
- `Tron.fromDecimal` was removed for *`toHex()`*.

Provider Access

In v2, `tron.currentProvider` was removed, in favor of `tron.providers`.

Disambiguating String Inputs

There are a number of places where an arbitrary string input might be either a byte-string that has been hex-encoded, or unicode characters in text. These are named `hexstr` and `text` in TronAPI. You specify which kind of `str` you have by using the appropriate keyword argument. See examples in *Type Conversions*.

In v1, some methods accepted a `str` as the first positional argument. In v2, you must pass strings as one of `hexstr` or `text` keyword arguments.

Notable methods that no longer accept ambiguous strings:

- *`sha3()`*
- *`toBytes()`*

Index