pycoalaip-bigchaindb Documentation

Release 0.0.4

BigchainDB

May 06, 2017

Contents

Important: Development Status: Alpha

Contents:

CHAPTER 1

pycoalaip-bigchaindb

BigchainDB ledger plugin for COALA IP's Python reference implementation.

- Development Status: Alpha
- Free software: Apache Software License 2.0
- Documentation: https://pycoalaip-bigchaindb.readthedocs.io.

Features

- Support for creating new keypair for use with BigchainDB
- Basic support for COALA IP entity creation

Credits

This package was created with Cookiecutter and the audreyr/cookiecutter-pypackage project template.

CHAPTER 2

Installation

Development release

To install pycoalaip-bigchaindb, run this command in your terminal:

```
$ pip install coalaip-bigchaindb
```

This is the preferred method to install pycoalaip-bigchaindb, as it will always install the most recent stable release.

If you don't have pip installed, this Python installation guide can guide you through the process.

From sources

The sources for pycoalaip-bigchaindb can be downloaded from the Github repo.

Clone the public repository:

```
$ git clone git://github.com/bigchaindb/pycoalaip-bigchaindb
```

And install it with:

```
$ python setup.py install
```

CHAPTER 3

Usage

To use pycoalaip-bigchaindb in a project:

```
import coalaip_bigchaindb
```

In most cases though, you won't be using this plugin directly but with pycoalaip. This plugin makes itself findable to `pycoalaip` through the `bigchaindb` entry point.

CHAPTER 4

Library Reference

Plugin

class coalaip_bigchaindb.**Plugin**(**nodes*)

BigchainDB ledger plugin for COALA IP's Python reference implementation.

Plugs in a BigchainDB instance as the persistence layer for COALA IP related actions.

__init__(**nodes*)

Initialize a *Plugin* instance and connect to one or more BigchainDB nodes.

Parameters ***nodes** (*str*) – One or more URLs of BigchainDB nodes to connect to as the persistence layer

generate_user()

Create a new public/private keypair for use with BigchainDB.

Returns

A dict containing a new user's public and private keys:

```
{
    'public_key': (str),
    'private_key': (str),
}
```

Return type dict

get_history(*persist_id*)

Get the transaction history of an COALA IP entity on BigchainDB.

Parameters **persist_id** (*str*) – Asset id of the entity on the connected BigchainDB instance

Returns

The ownership history of the entity, sorted starting from the beginning of the entity's history (i.e. creation). Each dict is of the form:

```
{
    'user': A dict holding only the user's public key
            (the private key is omitted as None).
    'event_id': The transaction id for the ownership event
}
```

Return type list of dict

Raises

- `coalaip.EntityNotFoundError` – If no asset whose id matches `persist_id` could be found in the connected BigchainDB instance
- `PersistenceError` – If any other unhandled error from the BigchainDB driver occurred.

get_status(*persist_id*)

Get the status of an COALA IP entity on BigchainDB.

Parameters **persist_id** (*str*) – Asset id of the entity on the connected BigchainDB instance

Returns

the status of the entity; one of:

```
'valid': the transaction has been written in a validated block
'invalid': the block the transaction was in was voted invalid
'undecided': the block the transaction is in is still undecided
'backlog': the transaction is still in the backlog
```

Return type str

Raises

- `coalaip.EntityNotFoundError` – If no transaction whose 'uuid' matches `persist_id` could be found in the connected BigchainDB instance
- `PersistenceError` – If any other unhandled error from the BigchainDB driver occurred.

is_same_user(*user_a*, *user_b*)

Check if `user_a` represents the same user as `user_b` on BigchainDB by comparing their public keys.

load(*persist_id*)

Load the data of the entity associated with the `persist_id` from BigchainDB.

Parameters **persist_id** (*str*) – Asset id of the entity being loaded on the connected BigchainDB instance

Returns The persisted data of the entity

Return type dict

Raises

- `coalaip.EntityNotFoundError` – If no asset whose id
- matches `persist_id` could be found in the connected
- BigchainDB instance
- `PersistenceError` – If any other unhandled error from the BigchainDB driver occurred.

save(*entity_data*, ***, *user*)
Create and assign a new entity with the given data to the given user's public key on BigchainDB.

Parameters

- **entity_data** (*dict*) – A dict holding the entity's data that will be saved in a new asset's asset definition
- **user** (*dict, keyword*) – The user to assign the created entity to on BigchainDB. A dict containing:

```
{
    'public_key': (str),
    'private_key': (str),
}
```

where 'public_key' and 'private_key' are the user's respective public and private keys.

Returns Asset id of the new entity

Return type str

Raises

- `coalaip.EntityCreationError` – If the creation transaction fails
- `PersistenceError` – If any other unhandled error from the BigchainDB driver occurred.

transfer(*persist_id*, *transfer_payload=None*, ***, *from_user*, *to_user*)
Transfer the entity matching the given `persist_id` from the current owner (`from_user`) to a new owner (`to_user`).

Parameters

- **persist_id** (*str*) – Asset id of the entity on the connected BigchainDB instance
- **transfer_payload** (*dict, optional*) – A dict holding the transfer's payload
- **from_user** (*dict, keyword*) – A dict holding the current owner's public key and private key (see *generate_user()*)
- **to_user** (*dict, keyword*) – A dict holding the new owner's public key and private key (see *generate_user()*)

Returns Id of the transaction transferring the entity from `from_user` to `to_user`

Return type str

Raises

- `coalaip.EntityNotFoundError` – If no asset whose id matches `persist_id` could be found in the connected BigchainDB instance
- `coalaip.EntityTransferError` – If the transfer transaction fails
- `PersistenceError` – If any other unhandled error from the BigchainDB driver occurred.

type
str – the type of this plugin (`'BigchainDB'`)

CHAPTER 5

About this Documentation

This section contains instructions to build and view the documentation locally.

If you do not have a clone of the repo, you need to get one.

Building the documentation

To build the docs, simply run

```
$ make docs
```

Viewing the documentation

You can either start a little web server locally, or open the HTML files with your browser.

To start a web server at http://localhost:5555/

```
# In project root, after making the docs
$ cd docs/_build/html/ && python -m SimpleHTTPServer 5555
```

Alternatively, open the *docs/_build/html/index.html* file in your web browser.

Making changes

Rebuild the docs and refresh the page on your web browser.

CHAPTER 6

Contributing

Contributions are welcome, and they are greatly appreciated! Every little bit helps, and credit will always be given.

You can contribute in many ways:

Types of Contributions

Report Bugs

Report bugs at https://github.com/bigchaindb/pycoalaip-bigchaindb/issues.

If you are reporting a bug, please include:

- Your operating system name and version.
- Any details about your local setup that might be helpful in troubleshooting.
- Detailed steps to reproduce the bug.

Fix Bugs

Look through the GitHub issues for bugs. Anything tagged with "bug" and "help wanted" is open to whoever wants to implement it.

Implement Features

Look through the GitHub issues for features. Anything tagged with "enhancement" and "help wanted" is open to whoever wants to implement it.

Write Documentation

pycoalaip-bigchaindb could always use more documentation, whether as part of the official pycoalaip-bigchaindb docs, in docstrings, or even on the web in blog posts, articles, and such.

Submit Feedback

The best way to send feedback is to file an issue at https://github.com/bigchaindb/pycoalaip-bigchaindb/issues.

If you are proposing a feature:

- Explain in detail how it would work.
- Keep the scope as narrow as possible, to make it easier to implement.
- Remember that this is a volunteer-driven project, and that contributions are welcome :)

Get Started!

Ready to contribute? Here's how to set up *pycoalaip-bigchaindb* for local development.

1. Fork the *pycoalaip-bigchaindb* repo on GitHub.
2. Clone your fork locally:

```
$ git clone git@github.com:your_name_here/pycoalaip-bigchaindb.git
```

3. Install your local copy into a virtualenv. Assuming you have virtualenvwrapper installed, this is how you set up your fork for local development:

```
$ mkvirtualenv pycoalaip-bigchaindb
$ cd pycoalaip-bigchaindb/
$ pip install -r requirements_dev.txt
```

4. Create a branch for local development:

```
$ git checkout -b name-of-your-bugfix-or-feature
```

   Now you can make your changes locally.

5. When you're done making changes, check that your changes pass flake8 and the tests, including testing other Python versions with tox:

```
$ flake8 coalaip_bigchaindb tests
$ pytest
$ tox
```

   To get flake8 and tox, just pip install them into your virtualenv.

6. Commit your changes and push your branch to GitHub:

```
$ git add .
$ git commit -m "Your detailed description of your changes."
$ git push origin name-of-your-bugfix-or-feature
```

7. Submit a pull request through the GitHub website.

Pull Request Guidelines

Before you submit a pull request, check that it meets these guidelines:

1. The pull request should include tests.
2. If the pull request adds functionality, the docs should be updated. Put your new functionality into a function with a docstring, and add the feature to the list in README.rst.
3. The pull request should work for Python 2.6, 2.7, 3.3, 3.4 and 3.5, and for PyPy. Check https://travis-ci.org/bigchaindb/pycoalaip-bigchaindb/pull_requests and make sure that the tests pass for all supported Python versions.

Tips

To run a subset of tests:

```
$ pytest tests.test_plugin
```

To run tests with debugging:

```
$ pytest -s
```

To run tests and break on errors:

```
$ pytest --pdb
```

CHAPTER 7

Credits

Development Lead

- BigchainDB <dev@bigchaindb.com>

Contributors

None yet. Why not be the first?

CHAPTER 8

History

0.0.3 (2017-05-05)

Update version of pycoalaip

0.0.4 (2017-05-05)

Ups, I forgot something

0.0.2 (2017-05-05)

Some changes during the OMI hackfest!

- Upgrade bigchaindb and driver's version to match up

0.0.1 (2017-02-17)

First alpha release on PyPI.

- **Upgraded to BigchainDB Server 0.9 (and along with it, bigchaindb-driver@v0.2)**
  - Users are now represented by public/private keypairs instead of verifying/signing keypairs
- Added support for querying ownership history of an entity
- Added support for entity transfers (and retransfers)
- Added `is_same_user()`

0.0.1.dev3 (2016-12-07)

- Implemented support for loading persisted Entities from BigchainDB
- Upgraded to bigchaindb-driver@v0.1.2

0.0.1.dev2 (2016-08-31)

- Fix packaging on PyPI

0.0.1.dev1 (2016-08-31)

- Development (pre-alpha) release on PyPI.

CHAPTER 9

Indices and tables

- genindex
- modindex
- search

Python Module Index

C

Index

Symbols

C

G

I

L

P

S

T