eInvoicing address registry Documentation

Release 0.1.0

Mikko Ohtamaa

January 30, 2017

Contents

Contents:

CHAPTER 1

Introduction

This project is a blockchain based e-invoicing address registry.

- EInvoicingRegistry smart contract in Solidity language
- Python based automated test suite
- HTML + JavaScript based interactive demo
- Free software: MIT license
- Documentation: https://eireg.readthedocs.io.

See also React based more complete front end

- Company self service portal project on Github

CHAPTER 2

Credits

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

CHAPTER 3

Background

3.1 Populus

Populus is a tool for the Ethereum blockchain and smart contract management. The project uses Populus internally. Populus is a Python based suite for

- Running arbitrary Ethereum chains (mainnet, testnet, private testnet)
- Running test suites against Solidity smart contracts

CHAPTER 4

Installation

4.1 Preface

Instructions are written in OSX and Linux in mind.

Experience needed

- Basic command line usage
- Basic Github usage
- Basic GNU make usage

4.2 Setting up - OSX

Packages needed

- Populus native dependencies

Get Solidity compiler. For OSX:

```
# Install solcjs using npm (JavaScript port of solc)
sudo npm install -g solc

# Symlink solcjs as solc, so that Populus finds it as default solc command
sudo ln -s `which solcjs` /usr/local/bin/solc
```

Clone this repository from Github.

Python 3.x required. See installing Python.

```
python3.5 --version
Python 3.5.2
```

Create virtualenv for Python package management in the project root folder (same as where `setup.py` is):

```
python3.5 -m venv venv
source venv/bin/activate
pip install -r requirements.txt
```

4.3 Setting up - Ubuntu Linux 14.04

Install dependencies:

```
sudo add-apt-repository ppa:fkrull/deadsnakes
sudo apt-get update
sudo apt-get install -y python3.5 python3.5-dev
sudo apt install -y git build-essential python3-setuptools libssl-dev
```

Install Go Ethereum:

```
sudo apt-get install software-properties-common
sudo add-apt-repository -y ppa:ethereum/ethereum
sudo apt-get update
sudo apt-get install -y ethereum solc
```

Then:

```
git clone git@github.com:nordledger/eireg.git
cd eireg
python3.5 -m venv --without-pip venv
source venv/bin/activate
curl https://bootstrap.pypa.io/get-pip.py | python
pip install -r requirements.txt
pip install -e .
```

CHAPTER 5

Usage

5.1 Working on a local private testnet

See a local private chain starts:

```
make run-local-test-chain
```

Abort with CTRL-C when it starts to generate DAG.

Copy in a custom keyfile for the coinbase account, so that we have the same deterministic coinbase account id for all runs:

```
cp keyfiles/UTC--2016-11-08T16-15-25.205056382Z--27d1755735abaf6cefb2299d18458b1091bb2c7b  chains/loc
```

Start a local private chain again and mine some ETH for a while (1 minute):

```
make run-local-test-chain
```

Abort with CTRL-C.

First we deploy a version of the contract on local chain managed by Populus.

```
make deploy-local
```

You will get deployment details:

```
Transaction Mined
=================
Tx Hash      : 0x3557ed87c7eb517c0e9c69dd15ba7d5c4064ce8d9b40caee40f3522fe6357a73
Address      : 0xb52fc9040759e04b793cbb094dc64ee051377c4c
Gas Provided : 362249
Gas Used     : 262249
```

Write down the deployed contract **Address** field. It varies across deployments.

This will take ~60 seconds. The default coinbase account is `0x27d1755735abaf6cefb2299d18458b1091bb2c7b`. It is configured in `populus.ini`.

5.2 Tieke / CSV import

Get Tieke electronic invoicing registry data dump as CSV.

Use `import-tieke-csv` tool to import existing records to a given smart contract address:

```
import-tieke-csv sample.csv local_test 0xb52fc9040759e04b793cbb094dc64ee051377c4c
```

Note: The local chain must not be running, but it is managed by this command.

Note: The tool lacks parallelism and is extremely slow at the moment.

5.3 Interacting with web browser

A simple interactive HTML demo is provided to interact with the contract.

Start geth deamon running a local chain. This is the same chain where we deployed the smart contract earlier:

```
make run-local-test-chain
```

It will start to mine transactions on your local computer.

In **another terminal** start a local development web server:

```
make run-web-server
```

Point your browser to:

```
http://localhost:8000
```

The demo directly interacts with Ethereum node over JSON-RPC protocol using web3.js library.

Fill in `Contract address` based on prior `populus deploy` command and **Connect** to contract.

5.4 Running automated test suite

To run test suite:

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:

6.1 Types of Contributions

6.1.1 Report Bugs

Report bugs at https://github.com/miohtama/eireg/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.

6.1.2 Fix Bugs

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

6.1.3 Implement Features

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

6.1.4 Write Documentation

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

6.1.5 Submit Feedback

The best way to send feedback is to file an issue at https://github.com/miohtama/eireg/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 :)

6.2 Get Started!

Ready to contribute? Here's how to set up *eireg* for local development.

1. Fork the *eireg* repo on GitHub.
2. Clone your fork locally:

```
$ git clone git@github.com:your_name_here/eireg.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 eireg
$ cd eireg/
$ python setup.py develop
```

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 eireg tests
$ python setup.py test or py.test
$ 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.

6.3 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/miohtama/eireg/pull_requests and make sure that the tests pass for all supported Python versions.

6.4 Tips

To run a subset of tests:

```
$ py.test tests.test_eireg
```

CHAPTER 7

Indices and tables

- genindex
- modindex
- search