Gardener Documentation

Release 0.1.0

Espeo Blockchain

Feb 04, 2019

Contents

This open source project solves the problem of getting knowledge from outside of the blockchain into smart contracts.

CHAPTER 1

Contents

Keyword Index, Search Page

1.1 Getting started

Requirements:

- `docker` installed and running
- `Node.js` >= 7.6 - `async/await` support

1. Download repositories: gardener-server, gardener-smart-contracts

```
git clone https://github.com/EspeoBlockchain/gardener-server.git
git clone https://github.com/EspeoBlockchain/gardener-smart-contracts.git
```

2. Copy smart contracts variables from template

```
cd gardener-smart-contracts
make copy-env
```

3. Copy server variables from template

```
cd ../gardener-server
make copy-env
```

4. Run test blockchain

```
make ganache
docker ps
```

5. Install smart contracts dependencies

```
cd ../gardener-smart-contracts
npm install
```

6. Migrate contracts to test blockchain

```
npx truffle migrate --network ganache --reset
```

7. Run server

```
cd ../gardener-server
make local
```

8. Make example oracle request

```
cd ../gardener-smart-contracts
npx truffle console --network ganache

truffle(ganache)> UsingOracle.deployed().then(instance => instance.request(
↪"json(https://api.coindesk.com/v1/bpi/currentprice.json).chartName"))
```

If you did everything correctly you should see something simmilar to

```
{ tx: '0x57a34e45e1f187ddeb4cbd1be3597561855563e5735a483a5b1edeb73a511278',
  receipt:
   { transactionHash:
↪'0x57a34e45e1f187ddeb4cbd1be3597561855563e5735a483a5b1edeb73a511278',
     transactionIndex: 0,
     blockHash: '0x212e264c92bef193e4531cc151d5b3b36818bc4bf82e154e84af6a7c6a153b43',
     blockNumber: 18,
     from: '0x90f8bf6a479f320ead074411a4b0e7944ea8c9c1',
     to: '0x9561c133dd8580860b6b7e504bc5aa500f0f06a7',
     gasUsed: 97604,
     cumulativeGasUsed: 97604,
     contractAddress: null,
     logs: [ [Object], [Object] ],
     status: '0x1',
     logsBloom:
↪'0x00040000000000000000100000000000000000000000000000000000000000000000000200000000000000010
↪',
     v: '0x1b',
     r: '0x21052fa282f9723d221ef288cec6e947cb2ba0ef3f1d470f5dc8845806f66977',
     s: '0x1723cb7b4288f6ae32f3495d666522859150fe3d0c8e4debd3a80d452f940f50' },
  logs:
   [ { logIndex: 1,
       transactionIndex: 0,
       transactionHash:
↪'0x57a34e45e1f187ddeb4cbd1be3597561855563e5735a483a5b1edeb73a511278',
       blockHash: '0x212e264c92bef193e4531cc151d5b3b36818bc4bf82e154e84af6a7c6a153b43
↪',
       blockNumber: 18,
       address: '0x9561c133dd8580860b6b7e504bc5aa500f0f06a7',
       type: 'mined',
       event: 'DataRequestedFromOracle',
       args: [Object] } ] }
```

9. Go to server container logs to check if response was sent.

1.2 Oracle theory

Oracle is a concept of getting information from outside of the blockchain to the smart contracts. Out of the box smart contracts cannot access anything outside of the blockchain network. That's where the oracle idea fits. The information exchange begins with the smart contract emitting an event describing the necessary information. A trusted off-chain server listening for such events parses it, gets data from a data source and passes it back to the smart contract.