Cyberscope

Documentation

Contract Deployment

June 2024

Table of Contents

Paxe Staking and Restaking contracts.sol

Staking and Restaking constructors

```
Staking.sol
constructor(
    PPAXE _pPAXE,
    address _PAXE,
    address _SAKAI,
    address _sakaiVaultV1,
    address _sakaiVaultV2,
    address _paxeTreasury,
    address _restakePool,
    address _paxeOracle,
    address _sakaiOracle,
    address _owner
) Ownable(_owner)

Restaking.sol
constructor(address _pPAXE, address _PAXE)
```

Constructor parameters that already exist

1. _PAXE (address)
2. _SAKAI (address)
3. _sakaiVaultV1 (address)
4. _sakaiVaultV2 (address)
5. _paxeTreasury (address)
6. _owner (address)

Arguments that will require new deployments

1. _pPAXE (ERC20 token)
2. _restakePool (Restaking.sol)
3. _paxeOracle (Oracle.sol)

4. _sakaiOracle (Oracle.sol)

_pPAXE (PPAXE.sol)

```
constructor(address newAdmin) ERC20("pPAXE Token", "pPAXE")
```

The only parameter is the *newAdmin*.

The admin will have minting rights, and he should also grant minting rights to *PaxeStaking.sol*. As such: grantRole(0x00000000000000000000000000000000000000000000000000000000000000000, address(Staking.sol))

As for *PaxeStandardStaking.sol*, the admin shall mint and send the amount of pPAXE he wants to distribute.

_restakePool (PaxeRestaking.sol)

```
constructor(address _pPAXE, address _PAXE)
```

The parameters are

1. _pPAXE (address)
2. _PAXE

Both are known, and it requires no further action in order to work. The reward token (PAXE) is sent here by Staking.sol when a user claims his rewards.

_paxeOracle/_sakaiOracle (Oracle.sol)

```
constructor(address factory, address tokenA, address tokenB)
```

The parameters are

1. factory (address) which is the UniswapV2/PancakeswapV2 factory contract
2. tokenA (address)
3. tokenB (address)

The oracle will use the tokens and factory to check the pair. If the pair is new or it doesn't have liquidity, the contract will revert. The reason it reverts is because it is unsafe to use so it needs more liquidity/more time for the pair to mature.

The Oracle needs to update every so often (done automatically at stake/claim) in order to update the cumulative prices so that its price is non-exploitable.

You will need to deploy 1 pair for PAXE and one pair for SAKAI.

For example on BSC Mainnet that would look like this:

```
SAKAI Oracle
new Oracle(0xcA143Ce32Fe78f1f7019d7d551a6402fC5350c73,
0x43B35e89d15B91162Dea1C51133C4c93bdd1C4aF,
0x55d398326f99059fF775485246999027B3197955)

PAXE Oracle
new Oracle(0xcA143Ce32Fe78f1f7019d7d551a6402fC5350c73,
0xd2a3eec06719d5ac66248003b5488e02165dd2fa,
0x55d398326f99059fF775485246999027B3197955)
```

Standard Staking PaxeStandardStaking.sol

Standard Staking Constructor

```
constructor(address _pPAXE, address _PAXE, address _paxeOracle, address
initialOwner) Ownable(initialOwner)
```

Constructor parameters that already exist

1. _PAXE (address)
2. initialOwner (address)

Arguments that will require new deployments

1. _pPAXE (ERC20 token)
2. _paxeOracle (Oracle.sol)

Adding a pool

```
function addPool(uint256 _apr, IERC20 _token, address oracle)
```

Function parameters

1. _apr (1 APR = 0.01% rewards per year, 1000 APR = 10% rewards per year)
2. _token (BTC/ETH/BNB etc)
3. oracle (the oracle of the _token given above)

Let’s have two examples.

Create one pool with 25% rewards per year, with BTC.

We could use the *Oracle.sol* used on *PaxeStaking.so*`, but since BTC has a V3 pool we should use the *OracleV3.sol*, which is better.

```
addPool(2500, 0x7130d2A12B9BCbFAe4f2634d864A1Ee1Ce3Ead9c,
address(OracleV3.sol))
```

Create one pool with 100% rewards per year, with WBNB.

We could use the *Oracle.sol* used on *PaxeStaking.so*`, but since WBNB has a V3 pool we should use the *OracleV3.sol*, which is better.

```
addPool(10000, 0xbb4CdB9CBd36B01bD1cBaEBF2De08d9173bc095c,
address(OracleV3.sol))
```

OracleV3.sol

```
constructor(address _pool)
```

The parameters are

1. _pool (address) The address of the V3 pool you want to create an Oracle for

The oracle will use the pool to fetch the tokens.

The OracleV3 does not need to update the cumulative prices in order to be a non-exploitable price oracle.

Let’s deploy one OracleV3 for BTC and one for WBNB.

For example on BSC Mainnet that would look like this:

```
BTC OracleV3
new OracleV3(0x46cf1cf8c69595804ba91dfdd8d6b960c9b0a7c4)

WBNB OracleV3
new OracleV3(0x36696169c63e42cd08ce11f5deebbcebae652050)
```

About Cyberscope

Cyberscope is a blockchain cybersecurity company that was founded with the vision to make web3.0 a safer place for investors and developers. Since its launch, it has worked with thousands of projects and is estimated to have secured tens of millions of investors' funds.

Cyberscope is one of the leading smart contract audit firms in the crypto space and has built a high-profile network of clients and partners.

The Cyberscope team

https://www.cyberscope.io