

# Oracle Deployment Guide

This section provides a comprehensive guide on deploying an oracle, including the nuances of script parameters and additional setup steps. Follow these steps to ensure accurate and reliable oracle deployment for price data retrieval.

## Step 1: Surveying DEX Liquidity

1. Identify DEXes with Sufficient Liquidity:
2. Begin by surveying the network for Decentralized Exchanges (DEXes) that offer sufficient liquidity. This ensures the oracle can retrieve reliable and accurate price data.

## Step 2: Selection of DEXes

1. Select Supported DEXes:
2. Choose DEXes that are supported by `SpotPriceAggregator`
3. or are forks of supported protocols. Supported DEXes can be found in `thecontracts/oracles/`
4. `directory` of the project.

## Step 3: Network Configuration

1. Configure the Network Settings:
2.
  - Skip this step if your network is supported. This can be checked by observing whether the network is mentioned (registered or not) during a test run, visible in the console output. This verification can be done also by reviewing `theregisterAll`
3.
  - method in the [Networks class](#)
4.
  - . If your network is listed there, it's considered supported, and no further action is required for registration in this step.
5.
  - Update the [Hardhat settings file](#)
6.
  - to configure the network.
7.
  - Utilize the `Networks`
8.
  - class from [solidity-utils](#)
9.
  - for network registration.
10.
  - Example configuration snippet:...
11.
  - `const`
12.
  - `{`
13.
  - `Networks`
14.
  - `}`
15.
  - `=`
16.
  - `require`
17.
  - `(`
18.
  - `'@1inch/solidity-utils/hardhat-setup'`
19.
  - `)`
20.
  - `;`
21.
  - `const`
22.
  - `net`
- 23.

```
24.     ◦ =
25.     ◦ new
26.     ◦ Networks
27.     ◦ (
28.     ◦ true
29.     ◦ ,
30.     ◦ 'mainnet'
31.     ◦ ,
32.     ◦ true
33.     ◦ )
34.     ◦ ;
35.     ◦ net
36.     ◦ .
37.     ◦ register
38.     ◦ (
39.     ◦ your_network_name
40.     ◦ ,
41.     ◦ networkId
42.     ◦ process
43.     ◦ .
44.     ◦ env
45.     ◦ .
46.     ◦ YOURNETWORK_RPC_URL
47.     ◦ ,
48.     ◦ process
49.     ◦ .
50.     ◦ env
51.     ◦ .
52.     ◦ YOURNETWORK_PRIVATE_KEY
53.     ◦ ,
54.     ◦ etherscan_network_name
55.     ◦ ,
56.     ◦ process
57.
```

```

58.     ◦ .
59.     ◦ env
60.     ◦ .
61.     ◦ YOURNETWORK_ETHERSCAN_KEY
62.     ◦ )
63.     ◦ ;
64.     ◦ const
65.     ◦ networks
66.     ◦ =
67.     ◦ net
68.     ◦ .
69.     ◦ networks
70.     ◦ ;
71.     ◦ const
72.     ◦ etherscan
73.     ◦ =
74.     ◦ net
75.     ◦ .
76.     ◦ etherscan
77.     ◦ ;
78.     ◦ ...

```

## Step 4: Environment Variables

1. Set Environment Variables:
2. Define necessary environment variables in the .env
3. file located at the project root. Include variables such as YOURNETWORK\_RPC\_URL
4. , YOURNETWORK\_PRIVATE\_KEY
5. , and YOURNETWORK\_ETHERSCAN\_KEY
6. with appropriate values:
7.
  - YOURNETWORK\_RPC\_URL
8.
  - : The RPC URL for accessing your network's node. This URL can support the HTTP header 'auth-key'. To use this header, append the header value to the URL using the
9.
  - symbol. For example: http://localhost:8545|HeaderValue
10.
  - . This format allows you to authenticate requests to your node.
11.
  - YOURNETWORK\_PRIVATE\_KEY
12.
  - : Your account's private key, which should be entered without the 0x
13.
  - prefix. This key is used for deploying contracts and executing transactions on the network.
14.
  - YOURNETWORK\_ETHERSCAN\_KEY
- 15.

- : The API key for an Etherscan-like blockchain explorer that supports your network. This key is necessary for verifying and publishing your contract's source code. Ensure you register for an API key with a compatible explorer service for your network.

## Step 5: Deploying Oracles

1. Deploy Oracles:
2.
  - Use the deploy script located at `deploy/commands/simple-deploy.js`
3.
  - . You can find a description of the script and how to use it in the [scripts description](#)
4.
  - .
5.
  - Configure the `PARAMS`
6.
  - object for each protocol you wish to deploy an oracle for. The parameters include:
    - \* `contractName`
7.
  - - : Name of the contract from the `contracts/oracles/`
8.
  - - `directory`.
9.
  - - `args`
10.
  - - : Arguments required by the contract (See contract's constructor).
11.
  - - `deploymentName`
12.
  - - : A name for your deployment, which will be used to create a file in the `deployments/`
13.
  - - `directory`.
14.
  - Ensure the `skip`
15.
  - [flag](#)
16.
  - is set to `false`
17.
  - to proceed with deployment.
18.
  - Example command for deployment: `yarn && yarn deploy`
19.
  - .

## Step 6: Deploying Wrappers

1. Deploy Wrappers:
2.
  - Follow similar steps as step 5 to deploy necessary wrappers and `MultiWrapper`
3.
  - . You can find different wrappers in the `contracts/wrappers/`
4.
  - `directory`. After `MultiWrapper`
5.
  - is deployed, it will be possible to edit these lists of wrappers.

## Step 7: Deploying OffchainOracle

1. Deploy OffchainOracle:
2.
  - Follow similar steps as step 5 to deploy the `OffchainOracle`

3.
  - . Make sure to include the deployed oracles (from step 5),MultiWrapper
4.
  - with wrappers (from step 6) and specifying the tokens you wish to use as connectors for price discovery.  
AfterOffchainOracle
5.
  - is deployed, it will be possible to edit these lists of oracles and connectors.[Edit this page](#) [Previous Examples](#)  
[Next Summary](#)