Internal Contract for price feed integration

InternalContract is the one that performs all the functions pertaining to blockchain connection, registering the feed's job, and initiating the jobs to fetch individual feed results. We have abstracted all those complexities under the hood, which is our internal contract, and created a Consumer contract to access the same. End users should be able to use the "Consumer contract" and access Oracle accordingly.

Internal contract gets data from Coinmarketcap for the pairs:- XDC and FXD against USD.

The internal contract needs to be "Approved" and "Deposited" before using it in actuality. We created a widget for the "Approval" & "Deposit", which can be downloaded from GitHub and the steps to install the widget with github link is given below.

Note: These feeds are setup in Plugin 1.0 version, as we are upgrading our network to Plugin 2.0, the contracts will be changed and the aggregator price value will be available. So please consider these for your testing and let us know if you want any changes to be applied

Consumer contract for price feed integration

Please note, as a user, you should have enough "#PLI" tokens to pay Oracle fee in order to access the price feed. For instance, the provided consumer contract needs to be deposited with a minimum 1 PLI to max 100. For this, we have provided a small widget right below.

Steps to use the widget

Step 1:

Clone the git package to your local pc

git clone **https://github.com/GoPlugin/paxo-plugin-integration-mainnet.git**

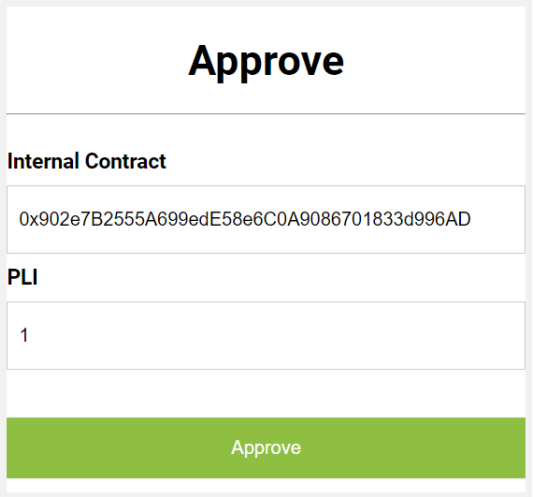
Step 2:

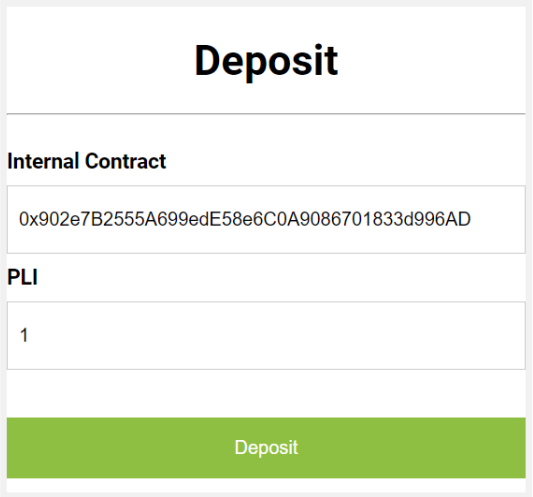
On the command line go to the **paxo-plugin-integration-mainnet/client** folder and do
npm install

Step 3:

On the command line initiate the widget using the below-mentioned command
npm start

Once the widget is installed, the server interface opens up through http://localhost:3000. The UI is given below for your understanding.





NOTE: Please use the **same wallet address** of your XDCPay for all your transactions.

Step 5:

In the widget Approve the "InternalContract" address **0x902e7B2555A699edE58e6C0A9086701833d996AD**, and then Deposit the "Internal Contract" using your PLI tokens from your wallet.

NOTE:

1) In the widget, the minimum PLI token you can approve and deposit is 1
2) In the widget, the maximum PLI token you can approve and deposit is 100

Based on the estimated usage you can deposit the PLI in between minimum and maximum, this PLI will be reserved in the InternalContract and whenever the end user uses this feed, the ORACLE FEE of 0.001 will be deducted.(Note, for production consumption this fee is subject to API subscription fee + infratraucture)

The consumer contract is given below for usage or you can get that in the repository under the consumer-contract folder. Make sure "CONTRACTADDR" variable matches with **0x902e7B2555A699edE58e6C0A9086701833d996AD,** which is the address of InternalContract deployed exclusively for Paxo.

Steps to be performed post-deployment:

Step 1:
Deploy ConsumerContract.sol using remix.

NOTE: Please use the **same wallet address** of your XDCPay that you used to approve, and deposit PLI in the widget.

```
pragma solidity ^0.4.24;


interface IInvokeOracle {
    function requestData(address _authorizedWalletAddress,string
fromIndex) external returns (uint256 requestId);

    function showPrice(uint256 _reqid) external view returns (uint256
answer, uint256 updatedOn);
}

contract ConsumerContract {
    address CONTRACTADDR = 0x902e7B2555A699edE58e6C0A9086701833d996AD;
    address private owner;
    mapping(string => uint256) latestRequestData;



    constructor() public{
        owner = msg.sender;
    }

    //Note, below function will not trigger if you do not put PLI in above
contract address
    function getPriceInfo(string  _symbol) external returns (uint256) {
        require(msg.sender==owner,"Only owner can trigger this");
        (latestRequestData[_symbol]) =
IInvokeOracle(CONTRACTADDR).requestData({_authorizedWalletAddress:owner,fro
mIndex:_symbol});
        return latestRequestData[_symbol];
    }
```

```
    //TODO - you can customize below function as you want, but below
function will give you the pricing value
    //This function will give you last stored value in the contract
    function show(string symbol) external view returns (uint256 _answer,
string _symbol, uint256 _timestamp) {
        (uint256 answer, uint256 updatedOn) =
IInvokeOracle(CONTRACTADDR).showPrice({_reqid: latestRequestData[symbol]});
        return (answer,symbol, updatedOn);
    }
}
```

Step 2:
Once ConsumerContract.sol is deployed, collect the deployed address. The wallet address with which you deployed the ConsumerContract should be sent to the Plugin team.

Plugin Team will authorize your ConsumerContract address, and your wallet address and reply back to your request.

Step 3:
Once you receive a successful reply from the Plugin Team, go to your ConsumerContract.sol deployment and click on the "**getPriceInfo"** function with the appropriate symbol. Give approx 20 to 30 seconds, for the jobs to get completed.

Step 4:
Click on the "**show**" by passing the symbol as parameter to get the value.

Note: to see the actual value for XDC/FXD divide the value obtained below by "1000000" (1 Million)

Reference image of customerContract deployment