OO Quick Start

0 to 1 Optimistic Oracle integration by example. The primary integration point into the UMA ecosystem is the Optimistic Oracle (OO). The OO is an oracle for arbitrary off-chain data

which leverages an interactive escalation game betweenrequesters ,proposers anddisputers and is secured byeconomic incentives .

This getting started tutorial will show you how to go from 0 to 1 with the Optimistic Oracle by executing the simplest possible request flow. Later in the docs you can find more information onhow the Optimistic Oracle works and dig deeper into its mechanism andmore sophisticated code examples .

If you prefer, you can also watch the following video tutorial in which we follow the step-by-step instructions described below.

?

A minimum viable Optimistic Oracle integration

You will be working through a simple smart contract that asks the oracle the question:Q:Did the temperature on the 25th of July 2022 in Manhattan NY exceed 35c? A:1 for yes. 0 for no. After submitting the request, you will propose a solution using the UMA Optimistic Oracle UI.

Once through liveness, you will fetch the price from the smart contract. This shows the full lifecycle of an Optimistic Oracle data request on the Görli testnet, interacting with an actual Optimistic Oracle contract deployment, without needing to write any code or clone any repos. It should give you the basic intuition as to how the Optimistic Oracle works without too much overhead and is a great starting point before digging deeper. Let's get started!

Prerequisites

To complete this tutorial you will need:

1. Metamask installed in a Chromium based browser (such asGoogle Chrome
2.) If you don't have it already, you can get Metamaskhere
3. .
4. A wallet set up in Metamask.
5. Görli
6. test ETH to send test transactions. You can get GETH from one of these faucetsTelegram Authenticated
7. ,Twitter Authenticated
8. ,Alchemy
9. ,slock.it
10. .
11.

Requesting data

First, we will work through the basic flow forasking the oracle for a piece of data . In this example, we are asking the Oracle for information on the weather but the request could be much more complex to could power any kind of smart contract system requiring data. Seeapproved price identifies and the sample projects for more inspiration on what is possible with the OO.

The contract used in this tutorial is meant to be a simple data request flow. The contract exposes a simplerequestData function which asks the OO a simple question about the weather.

1. Go to this example contract on Remix
2. . This gives you the minimum data request and retrieval flow possible. We'll work through the code in the sections that follow.
3. Click on "gist-fba..." to see the files in the gist, and clickOO_GettingStarted.sol
4. to open to Solidity file.
5. In the far left hand menu, click the link to deploy and run transactions (which looks like the Ethereum logo and a right arrow).
6. In the "Environment" dropdown, choose "Injected Provider," and connect to your wallet.Make sure you are connected to Görli within your metamask wallet!
7. You don't want to deploy to a real network and spend real gas, and the Görli Optimistic Oracle address hardcoded into the contract will not work on other networks.
8. Under the "Contract" dropdown, selectOO_GettingStarted
9. .
10. Click "Deploy" and confirm the transaction with your wallet.
11. You should now see theOO_GettingStarted
12. contract under "Deployed Contracts". Clicking the dropdown carrot will reveal buttons for each of the functions in the

contract.
13. ClickrequestData
14. to request the data specified in the contract's ancillary data, asking about the temperature in Manhattan on July 25th, 2022. Confirm the transaction in your wallet. This will submit a data request to the Optimistic Oracle.
15.

What we've done in the above steps is: a) deploy a smart contract and b) submit a "price request" to the Optimistic oracle through the call to the Optimistic Oracle'srequestPrice function.

Proposing data

Now that we've asked the OO a question, it's time to propose a solution! The Optimistic Oracle works via aninteractive escalation game wherein a) requesters ask questions b) proposers provide solutions and c) disputers monitor the proposals and dispute if they are invalid. If no dispute is done then the proposal is taken as valid. In this example, we'll be acting as the proposer who will be answering the question we asked in the previous section

1. Go to theGörli Optimistic Oracle UI
2. to see your request, which should be aYES_OR_NO_QUERY
3. at the top of the index page. Click on the request to go to the details page.
4. Click "Connect Wallet" and make sure you are connected to the Görli testnet so that you can make a proposal. You should see all the information posed in the data request: the requester, identifier, timestamp, ancillary data and a link to the UMIP.
5. Submit your proposal (either1
6. or0
7.). The actual value you submit is not super important in this case since this is just for testing purposes, but thecorrect response
8. is0
9. . In doing this we are acting as the "proposer" providing a solution.
10.

Once you've provided an answer to the question the proposal enters into a liveness period of 30 seconds. During this time it's up to Disputers to verify the correctness of what the proposer provided. Note that in a main net price request you'll normally set the liveness to much longer than 30 seconds (say two hours), the proposer will be bonded (if they propose wrong they lose money) and the proposer will be rewarded for providing a valid answer to the question.

Settling the final answer

Finally, we can fetch the data proposed in the previous step from the smart contract. For this example, we assume that there the proposed data was correct (was not disputed). Head back to your Remix tab and:

1. After the transaction finalizes, wait 30 seconds (the challenge window) and then clicksettleRequest
2. on theOO_GettingStarted
3. contract on Remix. Confirm the transaction in your wallet. Because the proposal was not disputed within the challenge window, it can now be settled.
4. ClickgetSettledPrice
5. to get the settled value. It should look likeint256: 0
6. undergetSettledPrice
7. if you proposed an answer of0
8. . You can also see details in the console logs.
9.

Congratulations! you've successfully integrated with the Optimistic Oracle, requested data, proposed data and fetched it within your smart contract! Note that the incentives in this toy example mean that there is no reason that anyone would provide thecorrect price (there was no reward for the proposers or disputes and there was no bond). A real world version of this shouldleverage custom bonds and more realistic liveness parameters.

Next Steps

Hopefully you got a basic understanding of the OO request flow from this getting started guide. Check out some of theexample tutorials where we walk through more details on the functions discussed in this guide.

Previous Optimistic Oracle v2 Next Deposit Box Last updated1 month ago On this page *A minimum viable Optimistic Oracle integration * Prerequisites * Requesting data * Proposing data * Settling the final answer * Next Steps

Was this helpful? Edit on GitHub