Fetch
ORACLE.COM

Decentralized Data You Can Trust.

Introduction

Fetch is a fully decentralized, transparent, and permissionless oracle system that is focused on securely placing data natively on PulseChain.

The oracle mechanism works by using simple crypto-economic incentives to secure data through staking and dispute mechanisms, while the community is bound by a token which utilizes anonymous governance system and monetary incentives to reward data reporters and development of the network.

The Oracle

Fetch pushes the horizon of the oracle far past arbitrary price data. The Fetch oracle is a protocol for answering on-chain any question of any format.

At a high level, Fetch is an oracle system where a bonded set of "reporters" answer questions on-chain for others to use freely. To create a properly incentivized system, Fetch mints a native token called "FETCH". Rewards in FETCH incentivize reporters to submit data using both peer-to-peer payments and inflationary rewards.

Using FETCH, parties can "tip" a specific question or "query" they want updated, then reporters can choose whether the reward for fetching the data is worth the cost of placing the value on-chain. The security of Fetch comes through a deposit of FETCH that acts as a bond or stake requirement in order for reporters to participate in providing data. The reporters risk losing this stake if they submit data that is successfully disputed.

Data Submission

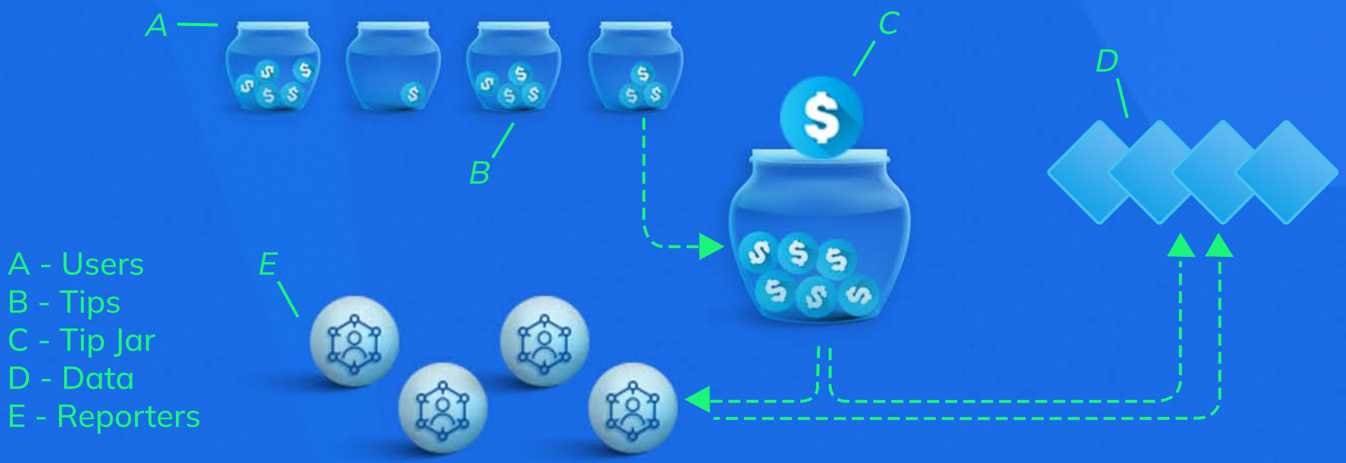
To become a reporter, an address will deposit the specified minimum bond amount for the chain. Those FETCH are locked (they act as a bond) until the reporter requests to withdraw them. The reporter then must wait one week before they can withdraw. For PulseChain it is 100 FETCH or $1500 USD in FETCH, whichever is greater in USD value. The minimum USD amount is implemented to provide users a minimum amount of security on the system.

Once the reporter has submitted their deposit, a reporter can submit values for any query (e.g. PLS/USD, Bitcoin block header information, weather data). After a reporter submits a value however, they must wait a certain time period (a configurable variable, starting at 12 hours), before submitting again. This is both to allow time for disputes as well as to increase the number of reporters in the system.

Reporters can choose to submit values for any ID they want, but in practice will likely pick the ID with the highest tip. ID's can be updated as frequently as they want. Reporters are rewarded in two ways:

- Tips
- Time-based inflationary rewards
- All bonded reporters get 2% of tips and time-based rewards

FIGURE 1



For parties needing data more frequently than when the time-based rewards are greater than the gas costs, they can simply add tips. The Fetch oracle can therefore be as fast as needed, parties will just need to pay for tips to cover expenses of the reporters.

The Data

Each request for data is given an ID on-chain, but specifications for the data are off-chain. Fetch uses a method for hashing a given query's specifications and the resulting hash is the bytes32 ID. This flexibility allows anyone to define a data specification and have a method for parsing the data for reporters to read.

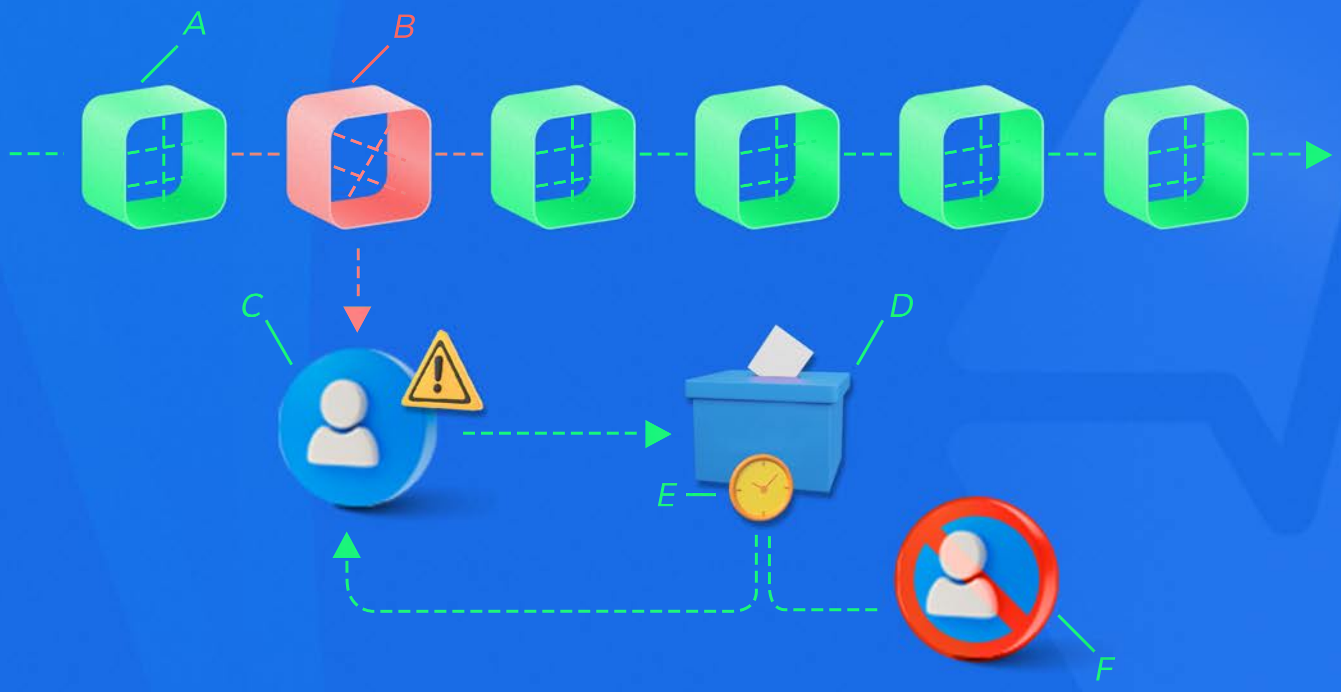
Data (returned values from reporters) are submitted as bytes, meaning any data type or number of variables can be pulled in a single query. An example of this type of data could be that a new request ID could be a BTC blockheader, a simple price (e.g. PLS/USD), or even an array of prices (e.g. [PLS price, BTC price, SPX price, VIX, EUR/USD]). The more data batching the system can provide in a query, the more efficient the system becomes (more values per transaction).

Reporters get to select which values they submit for. Some data requests may be very vague (e.g. the price of BTC/USD), but some may be more specific or manual (the rainfall in inches in Nairobi as measured by one website). Community members can propose (and tip on) new queries under the guide rails of Fetch's data specifications. These specifications will provide clarity as to data definition for those needing to verify the validity of the data. If reporters do not feel comfortable submitting or supporting a certain query (e.g. if you need a paid api feed), the reporter does not have to submit for it. Parties who wish to build reporter support for their query should follow best practices when selecting data for their query (publish data specification on github, promote/ educate in the community), but will also need to tip a higher amount to incentivize activity.

Disputes

Fetch data values can be used as soon as the data is placed on-chain, however the longer a user waits once the data is submitted on chain, the more probable it is to remain, and therefore be secure; assuming any value that remains on-chain is valid due to economic incentives to dispute invalid ones. Values are able to be disputed and taken off-chain for the same time frame as the reporter lock (12 hours).

FIGURE 2



A - Good Price B - Bad Price C - Disputer
D - Governance E - Time F - Slashed Reporter

Any party can challenge data submissions when a value is placed on-chain. A challenger must submit a dispute fee (10% of the bond amount) to each challenge. Once a challenge is submitted, the potentially malicious reporter (C in Figure 2) who submitted the value is placed in a locked state for the duration of the vote. For the next two days, the Fetch governance contract votes on the validity of the reported value (D in Figure 2). A proper submission is one that corresponds to a valid query as defined off-chain in the Fetch dataSpecs[1]. Although a correct answer should be known to the miners, the ambiguity (lack of an exact correctness in this case) of validity is a feature and corresponds to "correct" being at the discretion or interpretation of the Fetch community.

Dispute Rounds

The Fetch dispute mechanism allows for multiple rounds of disputes. After the first round, the cost of each subsequent dispute round increases exponentially:

$$disputeFee_{id,t,r>1} = disputeFee_i \times 2^{disputeRounds_{id,t}-1}$$

Where

$disputeFee_i$ is the initial dispute fee
$disputeRounds_{id,t}$ is the number of dispute rounds for a specific ID and timestamp

Dispute Resolution

At the end of the voting period, and if no new round is initiated, the votes are tallied. If found guilty, the malicious reporter's deposit goes to the disputing party; otherwise the fee paid by the disputer is given to the wrongly accused reporter.

Dispute Fees

The dispute fee is 10% of the bond amount.

[1] https://github.com/fetchoracle/dataSpecs

$$disputeFee_i = bondAmount/10$$

Where

bondAmount is the deposit required from each reporter to be able to provide data

If multiple disputes are performed on the same ID, a party might be trying to censor values by disputing good values. To counteract this, the dispute fee increases with each dispute on the same ID.

$$disputeFee_{id,t,r=1} = disputeFee_i \times 2^{disputeRounds_{id}-1}$$

Where

$disputeFee_i$ is the initial dispute fee
$disputeRounds_{id}$ is the number of disputes open for a specific ID

For this reason, it quickly becomes prohibitively expensive for a malicious party to simply dispute good values to censor a contract from reading them.

Replacement Data Tipping

Once a value is disputed it is taken off chain. For users who do not wish to wait for the result of the two day (or longer) vote, they can simply request the value again.

Invalid Data Query

Votes can be settled in one of three ways, true, false, or invalid. An invalid result means that the data is removed from the chain, but the reporter and the disputer do not lose tokens. An example of this would be a prediction market on who is the president the day after an election. If the election is not settled (the winner is unknown), but the oracle places a value on-chain before the result is known, it may end up being right, but at the time of the dispute/value submission, it isn't. This could be a case where the community decides to rule on the dispute as invalid. Leaving this option of ambiguity to the community affords the Fetch system more flexibility and reduces the chances that one of the two honest parties (a dispute and a reporter) are punished.

Governance

Fetch governance is used to settle disputes. The Fetch governance system aims to balance the voting power amongst various members of the community, specifically: holders, reporters, users and the team. FETCH stakeholders all want Fetch to continue to grow, but the approach and needs of each group can be different. Each of these stakeholders' categories are weighted equally. However, users and reporters can increase their voting share within their User and Reporter category, based on their system participation.

For Fetch, there are four groups of stakeholders identified in the Fetch system:

- FETCH holders
- Reporters
- Users
- Team

Within the FETCH holder category, holder weights are measured as the balance of FETCH on the chain where the vote is taking place. For reporters, each submitted data point affords them additional non-transferable voting power weighted at 1FETCH per successful submission. However, reporters must be actively bonded to be able to vote with their allocated non-transferable voting power. Users are weighted by the number of tips they have paid into the system. The Team's multi-sig wallet acts as a tie breaker and helps protect the integrity of the system in the early days but can and should be forked out as the network matures (when there are sufficient reporters, users and holders to provide checks and balances among themselves).

It is important to note that Fetch can be deployed on multiple chains. Although the main Fetch token is staying on the PulseChain network, users or the Fetch team can launch Fetch on any chain. However, the governance (dispute resolution) of that specific chain would be controlled by the stakeholders on that chain and inflationary time-based rewards only exist on PulseChain.

Monetary Policy

Fetch utilizes a fixed minting policy to incentivize reporters through time-based rewards and protocol improvements and maintenance. This can only be changed through a fork.

Security

Security is achieved through Fetch's architecture, which uses a simple bond/dispute mechanism to source correct values. Ultimate ownership and security in the system is afforded by our governance contract, which we aim to align incentives from holders, reporters, users and the team. There are two primary metrics used when determining the security of an oracle:

- How can a bad value get put on-chain? (e.g. BTC/USD is 10M)
- How much does it cost to censor the oracle? (no good values can get through)

For the former, this is the cost to break the governance contract of Fetch. If we assume that a bad value will get caught (not go through unnoticed), then the bad value would go to a dispute. For the value to be put back on-chain, the vote would need to settle such that an incorrect value is deemed correct. With multiple rounds of voting, the malicious party would need to get to the point where they have 51% of the voting power in the system and for this they would have to take over three of four equally weighted governance stakeholders (holders, users, reporters, and the team). Since the governance contract is only partially based on the token weight (token weighted FETCH holders, reporters, users), there are other factors at play.

Where VS = voting shares
VS = total supply + reporter votes + user(tipping) votes + team

Each stakeholder group is equally weighted. The outcome of each dispute vote can be true(for), false(against), or invalid. The percentage of votes for each outcome within each group is calculated against the total votes received per stakeholder group and then evaluated to determine the vote verdict.

$$\text{WeightedVote}_{outcome} = \frac{\text{FetchHolders}_{outcome}}{\sum \text{FetchHolder}} + \frac{\text{reporter}_{outcome}}{\sum \text{reporter}} + \frac{\text{user}_{outcome}}{\sum \text{user}} + \frac{\text{team}_{outcome}}{\sum \text{team}}$$

Where

outcome is true, false and invalid

After each *WeightedVote* is calculated for the possible outcomes, they are evaluated to determine the dispute outcome such that the final verdict is that which is greater than invalid and the other outcome.

$$\text{If } \text{WeightedVote}_{for} > \text{WeightedVote}_{against} + \text{WeightedVote}_{invalid} \Rightarrow \text{Passed(true)}$$

$$\text{If } \text{WeightedVote}_{against} > \text{WeightedVote}_{for} + \text{WeightedVote}_{invalid} \Rightarrow \text{Failed(false)}$$

Otherwise, the dispute is considered invalid.

If the system is newly deployed, there will be minimal reporter and user votes. Without the equal weighting among categories, the system would be a straight token weighted vote, with the cost to get 51% simply being 51% of the market cap or 51%. With equal weighting among stakeholders an attacker would need 51% of the market cap, tip once and report once.

In a system with reporter and user(tipper) votes, the attack vectors are different. The total number of reporter votes in the system is the count of all historical mining events among actively bonded reporters. At a rate of one mining event per minute (a very fast chain), this would indicate 525,000 reporter votes given each year. With a current supply of around 2M FETCH and about 1/10 that many mining events, this makes even a best-case scenario for the attacker prohibitively expensive to carry out in any manner that is not a multi-year attack.

To break the system via tipping, the malicious attacker could recycle the tips so as not to drain liquidity and increase the price. However, they would have to also take over two more stakeholder groups to be successful.

Overall, the cost to get a bad value on-chain is prohibitively high. Unless a party was breaking Fetch just for the fun of it, the much cheaper option is to simply censor Fetch for a certain period of time, as most use cases require some finality in the oracle (they will not wait for the entire Fetch system to settle a bad value dispute).

Since any value that is disputed will be put to a vote by all stakeholders the simple cost to censor is:

Cost of a stakeblock time of underlying system (since Fetch is as fast the underlying system)

As long as this value is higher than the cost to 51% attack a given chain (e.g. on PulseChain, to censor transactions at the miner level), Fetch should be considered a censorship resistant oracle for use on that network. With current costs at a 100 FETCH bond requirement or $1500 and a $15 price, if you assume even 10 second block times on PulseChain, the minimum cost to break is(it increases as the price of FETCH increases):

100 FETCH bond × $15/FETCH × (6×60) = ~$540,000/ hr

The cost to censor via disputes is:

*Cost to dispute = disputeFee * 2^ blocks per period*

Therefore the cost to dispute for even 10 minutes on a 30 second block chain (assuming the minimum 10 FETCH cost at a $15 price):

10 FETCH bond × $15/FETCH × 2^2×10 = ~$157M

Looking at our formula, we can summarize that security increases when:

- The share of those voting increases
- The price of the token increases

Additionally, a minimum threshold of reporters is also essential to the proper functioning of the Fetch system. The more parties that are available to submit data, the more decentralized our reporter set will be. An active and watchful community is also one of the big missing pieces in many protocols. Just because you are "theoretically" secure if there is an arbitrage opportunity, DeFi has seen many protocols exploited because those opportunities are left unfilled. It is the job of the Fetch ecosystem to properly incentivize and monitor the activity to make sure active diligence is being performed.

Use Cases

The new Fetch system, with its expansion to arbitrary data types, works as an oracle for any piece of offchain information. The specific structure of Fetch, with its lack of finality and ambiguously defined data points make it a unique oracle and one that is not fit for high speed values needing instant accuracy or trusted endpoints that are not open to any reporter.

That said, there are still a number of use cases that Fetch works well for and we look forward to expanding our user base and our community around these and even more creative use cases:

- Price feeds (e.g. PLS/LOAN TWAP)
- Prediction Markets (e.g. who is the current president of the United States?)
- Bridging assets (e.g. bring BTC block headers onto PulseChain) - L2
- Security (data availability, sequencer validation)