**Introduction**

Pyth Network is an oracle that publishes financial market data to multiple blockchains. Our market data is contributed by over 80 [first-party publishers](https://pyth.network/publishers/), including some of the biggest exchanges and market making firms in the world. We offer price feeds for a number of different asset classes, including [US equities, commodities, and cryptocurrencies](https://pyth.network/price-feeds/). Each price feed publishes a [robust aggregate](https://docs.pyth.network/design-overview/price-aggregation) of publisher prices that updates multiple times per second.

Price feeds are available on multiple blockchains and can also be used in off-chain applications. Pyth offers two sets of price feeds for different applications:

* ​[Solana Price Feeds](https://docs.pyth.network/solana-price-feeds) are available in mainnet for Solana
* ​[Pythnet Price Feeds](https://docs.pyth.network/pythnet-price-feeds) are available in mainnet for most [EVM chains](https://docs.pyth.network/pythnet-price-feeds/evm), including Ethereum, BNB, Avalanche, and more. These feeds are also available in [Aptos](https://docs.pyth.network/pythnet-price-feeds/aptos) mainnet and will be coming soon to Cosmos chains and other ecosystems.

Application developers can follow the links above to learn more about how to consume Pyth Network data for the relevant ecosystem.

Other useful links may be:

* ​[Publish Data](https://docs.pyth.network/publish-data) -- Data Providers can read these documents to learn how to publish data to the Pyth Network.
* ​[How Pyth Works](https://docs.pyth.network/design-overview) -- Learn more about the mechanisms behind the network.
* ​[Pyth Metrics](https://docs.pyth.network/metrics) -- See metrics of the network's adoption and growth.
* ​[Pyth Tutorials](https://youtube.com/playlist?list=PL-wxIsxi1V--5-p0eREKI0H8eszz7MGEg) -- Developers can watch contributors explaining the inner workings of the Pyth network and how to integrate with the Pyth Price Feeds in your application.

**Pythnet Price Feeds**

Pythnet price feeds use an on-demand price update model, where users are responsible for posting price updates on-chain when needed. Please see [*On-Demand Updates*](https://docs.pyth.network/pythnet-price-feeds/on-demand) to learn more about this model and what this means for integrators.

In the on-demand model, developers should integrate Pyth into both their on-chain and off-chain code:

1. 1.

On-chain programs should read prices from the Pyth program deployed on the same chain

1. 2.

Off-chain frontends and jobs should include Pyth price updates alongside (or within) their application-specific transactions.

Pyth provides ecosystem-specific SDKs to assist with both the on- and off-chain pieces of the integration. The easiest way to use Pyth price feeds is to integrate the appropriate SDKs into your application. Before getting started with an SDK, please read [*Using Price Feeds*](https://docs.pyth.network/pythnet-price-feeds/best-practices) to understand how Pyth price feeds are represented, and to learn best practices to use Pyth prices safely and correctly. Then, follow the links below to find the right SDK for your ecosystem:

# On-Demand Updates

Pyth Network uses an on-demand price update model that is slightly different from other oracles you may be more familiar with. Most oracles today use a push model, where the oracle runs an off-chain process that continuously sends transactions to update an on-chain price. In contrast, Pyth Network does not operate an off-chain process that pushes prices on-chain. Instead, it delegates this work to Pyth Network users. Pyth price updates are created on [Pythnet](https://docs.pyth.network/design-overview/pythnet) and streamed off-chain via the Wormhole Network, a cross-chain messaging protocol. These updates are signed such that the Pyth on-chain program can verify their authenticity. Updating the on-chain price is a permissionless operation: anyone can submit a valid Wormhole message to the Pyth contract to update the price. Typically, users of Pyth Network prices will submit a single transaction that simultaneously updates the price and uses it in a downstream application.

On-chain prices can only move forward in time. If a user submits a Wormhole message with a less recent price the Pyth program will not fail but will also not update the price. This in particular means that there's no guarantee that when a user atomically updates the price and then interacts with an application powered by Pyth, the price that the application will read will be equal to the price the user submitted.

You can find an in-depth explanation from one of our contributors, Jayant: [Explaining the Pyth Network On-Demand Oracle Model: Pyth Tutorials](https://youtu.be/qdwrs23Qc9g)​

## Advantages

The on-demand model has several benefits over the push model:

* **Gas efficiency** -- On-chain prices are only updated when they are needed. In the push model, the oracle can waste gas by submitting price updates that no one will use. Furthermore, the cost of updating the oracle is distributed amongst its users instead of borne entirely by Pyth Network. The cost of maintaining the on-chain prices can be substantial for a single entity, but is minimal when spread across all users. Many of the subsequent advantages follow from the fact that Pyth Network does not have to pay gas fees for every single update.
* **High update frequency** -- Pyth Network price feeds update once per second, which is faster than the blocktime of most blockchains. Such frequent updates would not be possible if every price had to be pushed on-chain. However, push oracles typically update even less frequently than the blocktime, because it is simply too expensive to update feeds more frequently.
* **Low latency** -- Every transaction can use a recent off-chain price, instead of relying on the last on-chain update pushed by the oracle itself.
* **More price feeds** -- Pyth Network can scale to thousands of price feeds due to its gas efficiency. The oracle incurs no added costs for each additional feed, and users pay gas costs for new feeds only when those feeds are used on-chain.
* **Reliable in volatile conditions** -- Push oracles can fail to land price updates in volatile market conditions. In these conditions, the oracle competes with other more valuable transactions (such as DEX trades or liquidations) for bandwidth, and often cannot pay enough to land price updates. This problem does not occur in the on-demand model, as Pyth price updates are incorporated into the valuable transactions themselves.
* **Common infrastructure** -- Every component of Pyth Network is shared across blockchains except for the contract deployed on the destination chain. These shared components can therefore be built with high reliability and accuracy targets, benefitting every chain the oracle is deployed on. This approach also allows Pyth Network to rapidly launch on new blockchains and ecosystems with all of the existing price feeds.
* **Sustainable** -- The Pyth Network protocol has been designed to allow for the optional enablement of data fees to update the state of an on-chain price feed. These fees will compensate data providers for their effort and motivates them to contribute additional data. Oracles without such a mechanism are inherently unsustainable and likely to fail if the operating organization runs out of money.

## Integration

Developers integrating with Pyth Network should build their application to submit the necessary price updates on their users' behalf. For example, if you need the BTC/USD price on-chain, then your frontend should submit the BTC/USD Pyth price update in every transaction that needs it. The Pyth Network SDKs cover both parts of this integration -- frontend and contract -- and are designed to simplify this process. These SDKs are blockchain-specific and are described in more detail in the subsequent sections of this documentation. Developers should also host an instance of the [price service](https://docs.pyth.network/pythnet-price-feeds/price-service), which is a convenient wrapper around the Wormhole Network that the frontend SDKs use to fetch on-demand updates.

## Fees

The Pyth Network protocol has been designed to allow for the optional enablement of data fees in order to update the state of an on-chain price feeds. The ongoing existence of and size of the fee will be determined by governance on a per-blockchain basis; until governance is live, the fee will be 1 of the smallest denomination of the blockchain's native token (e.g., 1 wei on Ethereum). The fees collected by the protocol will go toward compensating data providers and possibly other uses as determined by governance.

Note that protocols integrating with Pyth Network can pass these fees along to their users. Whenever a user submits a transaction that requires a price update, that transaction can also include payment of the necessary fee. This approach charges end users in proportion to their usage of Pyth Network data. The Pyth Network SDKs use this approach by default and include all of the necessary logic for computing and sending the fee along with every transaction.

In addition to update fees, end users ultimately bear the gas cost of updating the Pyth Network price feeds, which means that their transactions cost a little more than they would in the push model. However, the cost of a single price update is minimal, so the combined gas and update fee should only be a small portion of the overall transaction cost for the end user.

## Adversarial selection

On-demand price updates gives users of Pyth Network some ability to select which price to use in a transaction. This ability is highly circumscribed by various constraints: on-chain prices must move forward in time and cannot be from too far in the past. However, users can still chose any price update that satisfies these constraints. This ability is functionally equivalent to latency: it allows users to see the price in the future before using a price from the past.

The simplest way to guard against this attack vector is to incorporate a staleness check to ensure that the price used in a transaction is sufficiently recent. The Pyth Network SDKs include this check by default, where queries for the price will fail if the on-chain time differs from the price's timestamp by more than a threshold amount. The default threshold is set per-chain, but is typically around 1 minute. Highly latency-sensitive protocols may wish to reduce this threshold to a few seconds to better suit their needs. Please also see the section on [latency mitigations](https://docs.pyth.network/pythnet-price-feeds/best-practices#latency) for additional ideas on how latency-sensitive protocols can minimize the impact of oracle latency.