Chainlink Data Streams

Mainnet Access

Chainlink Data Streams is available on Arbitrum Mainnet and Arbitrum Sepolia.

Talk to an expert

Contact us to talk to an expert about integrating Chainlink Data Streams with your applications.

Chainlink Data Streams provides low-latency delivery of market data offchain that you can verify onchain. With Chainlink Data Streams, decentralized applications (dApps) now have on-demand access to high-frequency market data backed by decentralized and transparent infrastructure. When combined withChainlink Automation , Chainlink Data Streams allows decentralized applications to automate trade execution, mitigate frontrunning, and limit bias or adverse incentives in executing non-user-triggered orders.

Traditional push-based oracles provide regular updates onchain when certain price thresholds or update time periods have been met. Chainlink Data Streams is built using a new pull-based oracle design that maintains trust-minimization using onchain verification.

Use cases

Pull-based oracles allow decentralized applications to access data that is updated at a high frequency and delivered with low latency, which enables several new use cases:

- Perpetual Futures:Low-latency data and frontrunning prevention enable onchain perpetual futures protocols that can compete on performance with centralized exchanges while still using more transparent and decentralized infrastructure.
- Options:Pull-based oracles allow timely and precise settlement of options contracts. Additionally, Data Streams provides more detailed market liquidity data that can support dynamic onchain risk management logic.
- Prediction Markets:Higher frequency data updates allow for applications where users can act quickly in response to real-time events and be confident in the accuracy of the data used in the settlement.

Billing

Chainlink Data Streams supports fee payments in LINK and in alternative assets, which currently includes native blockchain gas tokens and their ERC20-wrapped version. Payments made in alternative assets have a surcharge when compared to LINK payments. You pay to verify reports from Data Streams onchain using the verifier contract. The price of verification depends on the stream ID that you are verifying. You pay per report verified. If you verify multiple reports in a batch, you pay for all of the reports included in that batch.

Contact us to learn more about Mainnet pricing.

Architecture

Chainlink Data Streams has the following core components:

- A Chainlink Decentralized Oracle Network (DON):This DON operates similarly to the DONs that powerChainlink Data Feeds , but the key difference is that it signs and delivers reports to the Chainlink Data Engine rather than delivering answers onchain directly. This allows the Data Streams DON to deliver reports more frequently for time-sensitive applications. Nodes in the DON retrieve data from many different data providers, reach a concensus about the median price of an asset, sign a report including that data, and deliver the report to the Chainlink Data Engine.
- The Chainlink Data Engine:The Data Engine stores the signed reports and delivers them to Chainlink Automation when it is requested.
- The Chainlink Verifier Contract:This contract verifies the signature from the DON to cryptographically guarantee that the report has not been altered from the time that the DON reached concensus to the point where you use the data in your application.

Using Chainlink Automation with Data Streams automates trade execution and mitigates frontrunning by executing the transaction before the data is recorded onchain. Chainlink Automation executes transactions only in response to the data and the verified report, so the transaction is executed correctly and independently from the decentralized application itself.

Comparison to push-based oracles

Chainlink's push-based oracles provide regular updates onchain. Chainlink Data Streams operates as a pull-based oracle where you can retrieve the data in a report and use it onchain any time. Verifying the report onchain confirms that the data was agreed upon and signed by the DON. While many applications benefit from push-based oracles and require data only

after it has been verified onchain, some applications require access to data that is updated at a higher frequency and delivered with lower latency. Pull-based oracles deliver these benefits while still cryptographically signing the data to ensure its veracity.

Additionally, pull-based oracles deliver data onchain more efficiently by retrieving and verifying the data only when the application needs it. For example, a decentralized exchange might retrieve a Data Streams report and verify the data onchain when a user executes a trade. A push-based oracle repeatedly delivers data onchain even when that data is not immediately required by users.

Example trading flow

One example of how to use Data Streams is in a decentralized exchange. An example flow might work using the following process:

1. A user initiates a trade by confirming aninitiateTradetransaction in their wallet.
2. The onchain contract for the decentralized exchange responds by emitting a Log Trigger event.
3. The Chainlink Automation upkeep monitors the contract for the event. When Automation detects the event, it runs thecheckLogfunction specified in the upkeep contract. The upkeep is defined by the decentralized exchange.
4. ThecheckLogfunction uses arevertwith a custom error calledStreamsLookup. This approach aligns withEIP-3668 and conveys the required information through the data in therevertcustom error.
5. Chainlink Automation monitors theStreamsLookupcustom error that triggers Chainlink Data Streams to process the offchain data request. Chainlink Data Streams then returns the requested signed report in thecheckCallbackfunction for Chainlink Automation.
6. Chainlink Automation passes the report to the Automation Registry, which executes theperformUpkeepfunction defined by the decentralized exchange. The report is included as a variable in theperformUpkeepfunction.
7. TheperformUpkeepfunction calls theverifyfunction on the Data Streams onchain verifier contract and passes the report as a variable.
8. The verifier contract returns averifierResponsebytes value to the upkeep.
9. If the response indicates that the report is valid, the upkeep executes the user's requested trade. If the response is invalid, the upkeep rejects the trade and notifies the user.

This is one example of how you can combine Data Streams and Chainlink Automation, but the systems are highly configurable. You can write your own log triggers to initiate upkeeps on Chainlink Automation for a various array of events. You can configure theStreamsLookupto retrieve multiple reports. You can configure theperformUpkeepfunction to perform a wide variety of actions using the report.

Read theGetting Started guide to learn how to build your own smart contract that retrieves reports from Chainlink Data Streams.