Chainlink Data Feeds

Talk to an expert

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

Chainlink Data Feeds are the quickest way to connect your smart contracts to the real-world data such as asset prices, reserve balances, NFT floor prices, and L2 sequencer health.

If you already started a project and need to integrate Chainlink, you can add Chainlink to your existing project with the @chainlink/contracts NPM package .

Types of data feeds

Data feeds provide many different types of data for your applications.

- Price Feeds
- Proof of Reserve Feeds
- NFT Floor Price Feeds
- Rate and Volatility Feeds
- L2 sequencer uptime feeds

Price Feeds

Smart contracts often act in real-time on data such as prices of assets. This is especially true in DeFi .

For example, Synthetix uses Data Feeds to determine prices on their derivatives platform. Lending and borrowing platforms like AAVE use Data Feeds to ensure the total value of the collateral.

Data Feeds aggregate many data sources and publish them onchain using a combination of the Decentralized Data Model and Offchain Reporting .

To learn how to use Price Feeds, see the Price Feeds documentation.

See the Data Feeds Contract Addresses page for a list of available networks and addresses.

Proof of Reserve Feeds

Proof of Reserves feeds provide the status of reserves for stablecoins, wrapped assets, and real world assets. Proof of Reserve Feeds operate similarly to Price Feeds, but provide answers in units of measurement such as ounces (oz) or number of tokens.

To learn more about Proof of Reserve Feeds, see the Proof of Reserve documentation.

See the Proof of Reserve Contract Addresses page for a list of available networks and addresses.

NFT Floor Price Feeds

NFT Floor Price Feeds provide the price of the lowest priced NFT available in a collection. These feeds operate similarly to Price Feeds so you can use the AggregatorV3Interface.sol contract to retrieve price answers.

To learn more, see the NFT Floor Price Feeds documentation.

See the NFT Floor Price Contract Addresses page for a list of available networks and addresses.

Rate and Volatility Feeds

Several feeds provide interest rate curve data, APY data, and realized asset price volatility.

To learn more, see the Rate and Volatility Feeds documentation.

See the Rate and Volatility Contract Addresses page for a list of available networks and addresses.

L2 sequencer uptime feeds

L2 sequencer feeds track the last known status of the sequencer on an L2 network at a given point in time. This helps you prevent mass liquidations by providing a grace period to allow customers to react to these events.

To learn how to use L2 sequencer uptime feeds feeds, see the L2 Sequencer Uptime Feeds documentation.

Components of a data feed

Data Feeds are an example of a decentralized oracle network and include the following components:

- Consumer: A consumer is an onchain or offchain application that uses Data Feeds. Consumer contracts use theAggregatorV3Interface to call functions on the proxy contract and retrieve information from the aggregator contract. For a complete list of functions available in theAggregatorV3Interface, see theData Feeds API Reference .
- Proxy contract: Proxy contracts are onchain proxies that point to the aggregator for a particular data feed. Using proxies enables the underlying aggregator to be upgraded without any service interruption to consuming contracts. Proxy contracts can vary from one data feed to another, but theAggregatorProxy.solcontract on Github is a common example.
- Aggregator contract: An aggregator is a contract that receives periodic data updates from the oracle network. Aggregators store aggregated data onchain so that consumers can retrieve it and act upon it within the same transaction. For a complete list of functions and variables available on most aggregator contracts, see theData Feeds API Reference .

To learn how to create a consumer contract that uses an existing data feed, read theUsing Data Feeds documentation.

Reading proxy and aggregator configurations

Because the proxy and aggregator contracts are all onchain, you can see the current configuration by reading the variables through anABI or using a blockchain explorer for your network. For example, you can see theBTC/USD proxy configuration on the Ethereum network using Etherscan.

If you read the BTC/USD proxy configuration, you can query all of the functions and variables that are publicly accessible for that contract including theaggregatoraddress,latestRoundData()function,latestAnswervariable,owneraddress,latestTimestampvariable, and several others. To see descriptions for the proxy contract variables and functions, see the source code for your specific data feed onEtherscan .

The proxy contract points to an aggregator. This allows you to retrieve data through the proxy even if the aggregator is upgraded. If you view theaggregatoraddress defined in the proxy configuration, you can see the aggregator and its configuration. For example, see theBTC/USD aggregator contract in Etherscan. This contract includes several variables and functions, including anotherlatestRoundData(). To see descriptions for the aggregator variables and functions, see the source code onGitHub orEtherscan .

You can call thelatestRoundData()function directly on the aggregator, but it is a best practice to use the proxy instead so that changes to the aggregator do not affect your application. Similar to the proxy contract, the aggregator contract has alatestAnswervariable,owneraddress,latestTimestampvariable, and several others.

Components of an aggregator

The aggregator contract has several variables and functions that might be useful for your application. Although aggregator contracts are similar for each data feed, some aggregators have different variables. Use thetypeAndVersion()function on the aggregator to identify what type of aggregator it is and what version it is running.

Always check the contract source code and configuration to understand how specific data feeds operate. For example, theaggregator contract for BTC/USD on Arbitrum is different from the aggregators on other networks.

For examples of the contracts that are typically used in aggregator deployments, see theocr repository on GitHub.

For a complete list of functions and variables available on most aggregator contracts, see theData Feeds API Reference .

Updates to proxy and aggregator contracts

To accommodate the dynamic nature of offchain environments, Chainlink Data Feeds are updated from time to time to add new features and capabilities as well as respond to externalities such as token migrations, protocol rebrands, extreme market events, and upstream issues with data or node operations.

These updates include changes to the aggregator configuration or a complete replacement of the aggregator that the proxy uses. If you consume data feeds through the proxy, your applications can continue to operate during these changes.

Proxy and aggregator contracts all have anowneraddress that has permission to change variables and functions. For example, if you read theBTC/USD proxy contract in Etherscan, you can see theowneraddress. This address is amulti-signature safe (multisig) that you can also inspect.

If youview the multisig contract in Etherscan using theRead as Proxyfeature, you can see the full details of the multisig including the list of addresses that can sign and the number of signers required for the multisig to approve actions on any

contracts that it owns.

The multisig-coordinated upgradability of Chainlink Data Feeds involves time-tested processes that balance collusion-resistance with the flexibility required to implement improvements and swiftly react to external conditions. The approach taken to upgradability will continue to evolve over time to meet user requirements.

Monitoring data feeds

When you build applications and protocols that depend on data feeds, include monitoring and safeguards to protect against the negative impact of extreme market events, possible malicious activity on third-party venues or contracts, potential delays, and outages.

Create your own monitoring alerts based on deviations in the answers that data feeds provide. This will notify you when potential issues occur so you can respond to them.

Check the latest answer against reasonable limits

The data feed aggregator includes both`minAnswer`and`maxAnswer`values . On most data feeds, these values are no longer used and they do not stop your application from reading the most recent answer. For monitoring purposes, you must decide what limits are acceptable for your application.

Configure your application to detect when the reported answer is close to reaching reasonable minimum and maximum limits so it can alert you to potential market events. Separately, configure your application to detect and respond to extreme price volatility or prices that are outside of your acceptable limits.

Check the timestamp of the latest answer

Chainlink Data Feeds do not provide streaming data. Rather, the aggregator updates its`latestAnswer`when the value deviates beyond a specified threshold or when the heartbeat idle time has passed. You can find the heartbeat and deviation values for each data feed at data.chain.link or in the Contract Addresses lists.

Your application should track the`latestTimestamp`variable or use the`updatedAt`value from the`latestRoundData()`function to make sure that the latest answer is recent enough for your application to use it. If your application detects that the reported answer is not updated within the heartbeat or within time limits that you determine are acceptable for your application, pause operation or switch to an alternate operation mode while identifying the cause of the delay.

During periods of low volatility, the heartbeat triggers updates to the latest answer. Some heartbeats are configured to last several hours, so your application should check the timestamp and verify that the latest answer is recent enough for your application.

Users should build applications with the understanding that data feeds for wrapped or liquid staking assets might have different heartbeat and deviation thresholds than that of the underlying asset. Heartbeat and deviation thresholds can also differ for the same asset across different blockchains. Combining data from multiple feeds, even those with a common denominator, might result in a margin of error that users must account for in their risk mitigation practices.

To learn more about the heartbeat and deviation threshold, read the Decentralized Data Model page.