**Chainlink Data Feeds**

**TALK TO AN EXPERT**

[Contact us](https://chain.link/contact?ref_id=DataFeed) 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](https://docs.chain.link/resources/create-a-chainlinked-project?parent=dataFeeds#installing-into-existing-projects) with the [@chainlink/contracts NPM package](https://www.npmjs.com/package/@chainlink/contracts" \t "_blank).

**Types of data feeds**

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

* [Price Feeds](https://docs.chain.link/data-feeds#price-feeds)
* [Proof of Reserve Feeds](https://docs.chain.link/data-feeds#proof-of-reserve-feeds)
* [NFT Floor Price Feeds](https://docs.chain.link/data-feeds#nft-floor-price-feeds)
* [Rate and Volatility Feeds](https://docs.chain.link/data-feeds#rate-and-volatility-feeds)
* [L2 sequencer uptime feeds](https://docs.chain.link/data-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](https://defi.chain.link/).

For example, [Synthetix](https://www.synthetix.io/) uses Data Feeds to determine prices on their derivatives platform. Lending and borrowing platforms like [AAVE](https://aave.com/) use Data Feeds to ensure the total value of the collateral.

Data Feeds aggregate many data sources and publish them on-chain using a combination of the [Decentralized Data Model](https://docs.chain.link/architecture-overview/architecture-decentralized-model?parent=dataFeeds) and [Off-Chain Reporting](https://docs.chain.link/architecture-overview/off-chain-reporting?parent=dataFeeds).

To learn how to use Price Feeds, see the [Price Feeds](https://docs.chain.link/data-feeds/price-feeds) documentation.

See the [Data Feeds Contract Addresses](https://docs.chain.link/data-feeds/price-feeds/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](https://docs.chain.link/data-feeds/proof-of-reserve) documentation.

See the [Proof of Reserve Contract Addresses](https://docs.chain.link/data-feeds/proof-of-reserve/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](https://docs.chain.link/data-feeds/price-feeds) so you can use the [AggregatorV3Interface.sol](https://github.com/smartcontractkit/chainlink/blob/master/contracts/src/v0.8/interfaces/AggregatorV3Interface.sol) contract to retrieve price answers.

To learn more, see the [NFT Floor Price Feeds](https://docs.chain.link/data-feeds/nft-floor-price) documentation.

See the [NFT Floor Price Contract Addresses](https://docs.chain.link/data-feeds/nft-floor-price/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](https://docs.chain.link/data-feeds/rates-feeds) documentation.

See the [Rate and Volatility Contract Addresses](https://docs.chain.link/data-feeds/rates-feeds/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](https://docs.chain.link/data-feeds/l2-sequencer-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 on-chain or off-chain application that uses Data Feeds. Consumer contracts use the [AggregatorV3Interface](https://github.com/smartcontractkit/chainlink/blob/master/contracts/src/v0.8/interfaces/AggregatorV3Interface.sol) to call functions on the proxy contract and retrieve information from the aggregator contract. For a complete list of functions available in the AggregatorV3Interface, see the [Data Feeds API Reference](https://docs.chain.link/data-feeds/api-reference/#aggregatorv3interface).
* **Proxy contract**: Proxy contracts are on-chain 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 the [AggregatorProxy.sol contract](https://github.com/smartcontractkit/chainlink/blob/develop/contracts/src/v0.7/dev/AggregatorProxy.sol" \t "_blank) 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 on-chain 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 the [Data Feeds API Reference](https://docs.chain.link/data-feeds/api-reference/#accesscontrolledoffchainaggregator).

To learn how to create a consumer contract that uses an existing data feed, read the [Using Data Feeds](https://docs.chain.link/data-feeds/price-feeds) documentation.

**Reading proxy and aggregator configurations**

Because the proxy and aggregator contracts are all on-chain, you can see the current configuration by reading the variables through an [ABI](https://docs.soliditylang.org/en/latest/abi-spec.html) or using a blockchain explorer for your network. For example, you can see the [BTC/USD proxy configuration](https://etherscan.io/address/0xF4030086522a5bEEa4988F8cA5B36dbC97BeE88c#readContract) 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 the aggregator address, latestRoundData() function, latestAnswer variable, owner address, latestTimestamp variable, and several others. To see descriptions for the proxy contract variables and functions, see the source code for your specific data feed on [Etherscan](https://etherscan.io/address/0xF4030086522a5bEEa4988F8cA5B36dbC97BeE88c" \l "code#L568" \t "_blank).

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 the aggregator address defined in the proxy configuration, you can see the aggregator and its configuration. For example, see the [BTC/USD aggregator contract](https://etherscan.io/address/0xF4030086522a5bEEa4988F8cA5B36dbC97BeE88c#code) in Etherscan. This contract includes several variables and functions, including another latestRoundData(). To see descriptions for the aggregator variables and functions, see the source code on [GitHub](https://github.com/smartcontractkit/libocr/blob/master/contract/AccessControlledOffchainAggregator.sol) or [Etherscan](https://etherscan.io/address/0xae74faa92cb67a95ebcab07358bc222e33a34da7" \l "code#F1#L1" \t "_blank).

You can call the latestRoundData() 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 a latestAnswer variable, owner address, latestTimestamp variable, 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 the typeAndVersion() 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, the [aggregator contract for BTC/USD on Arbitrum](https://arbiscan.io/address/0x942d00008d658dbb40745bbec89a93c253f9b882#code) is different from the aggregators on other networks.

For examples of the contracts that are typically used in aggregator deployments, see the [libocr repository](https://github.com/smartcontractkit/libocr/blob/master/contract/" \t "_blank) on GitHub.

For a complete list of functions and variables available on most aggregator contracts, see the [Data Feeds API Reference](https://docs.chain.link/data-feeds/api-reference/#accesscontrolledoffchainaggregator).

**Updates to proxy and aggregator contracts**

To accommodate the dynamic nature of off-chain 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 an owner address that has permission to change variables and functions. For example, if you read the [BTC/USD proxy contract](https://etherscan.io/address/0xF4030086522a5bEEa4988F8cA5B36dbC97BeE88c#readContract) in Etherscan, you can see the owner address. This address is a [multi-signature safe](https://docs.gnosis-safe.io/introduction/the-programmable-account/gnosis-safe) (multisig) that you can also inspect.

If you [view the multisig contract](https://etherscan.io/address/0x21f73d42eb58ba49ddb685dc29d3bf5c0f0373ca#readProxyContract) in Etherscan using the *Read as Proxy* feature, 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](https://github.com/smartcontractkit/libocr/blob/9e4afd8896f365b964bdf769ca28f373a3fb0300/contract/AccessControlledOffchainAggregator.sol#L33). These variables prevent the aggregator from updating the latestAnswer outside the agreed range of acceptable values, but they do not stop your application from reading the most recent answer.

Configure your application to detect when the reported answer is close to reaching minAnswer or maxAnswer and issue an alert so you can respond to a potential market event. 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](https://data.chain.link/" \t "_blank) or in the [Contract Addresses](https://docs.chain.link/data-feeds/price-feeds/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](https://docs.chain.link/architecture-overview/architecture-decentralized-model?parent=dataFeeds#aggregator) page.