Consuming Data Feeds

You can use Chainlink Data Feeds to connect your smart contracts to asset pricing data like theETH / USD feed. These data feeds use data aggregated from many independent Chainlink node operators. Each price feed has an onchain address and functions that enable contracts to read pricing data from that address.

This guide shows you how to read Data Feeds and store the value onchain using Solidity. To learn how to read feeds offchain or use different languages, see theUsing Data Feeds on EVM Chainsguide. Alternatively, you can also learn how to use Data Feeds onSolanaorStarkNet.

The code for reading Data Feeds on Ethereum or other EVM-compatible blockchains is the same for each chain and each Data Feed types. You choose different types of feeds for different uses, but the request and response format are the same. The answer decimal length and expected value ranges might change depending on what feed you use.

Using Data Feeds on L2 networks

If you are using Chainlink Data Feeds on L2 networks like Arbitrum, Optimism, and Metis, you must also check the latest answer from the L2 Sequencer Uptime Feed to ensure that the data is accurate in the event of an L2 sequencer outage. See theL2 Sequencer Uptime Feedspage to learn how to use Data Feeds on L2 networks.

Before you begin

If you are new to smart contract development, learn how toDeploy Your First Smart Contractbefore you start this guide.

Examine the sample contract

This example contract obtains the latest price answer from theBTC / USD feedon the Sepolia testnet, but you can modify it to read any of the differentTypes of Data Feeds.

// SPDX-License-Identifier: MITpragmasolidity^0.8.7;import{AggregatorV3Interface}from"@chainlink/contracts/src/v0.8/shared/interfaces/AggregatorV3Interface.sol";/ *** THIS IS AN EXAMPLE CONTRACT THAT USES HARDCODED * VALUES FOR CLARITY. * THIS IS AN EXAMPLE CONTRACT THAT USES UN-AUDITED CODE. * DO NOT USE THIS CODE IN PRODUCTION. ***// * If you are reading data feeds on L2 networks, you must * check the latest answer from the L2 Sequencer Uptime * Feed to ensure that the data is accurate in the event * of an L2 sequencer outage. See the * https://docs.chain.link/data-feeds/l2-sequencer-feeds * page for details. */contractDataConsumerV3{AggregatorV3InterfaceinternaldataFeed;/ * Network: Sepolia * Aggregator: BTC/USD * Address: 0x1b44F3514812d835EB1BDB0acB33d3fA3351Ee43 */*constructor() {dataFeed=AggregatorV3Interface(0x1b44F3514812d835EB1BDB0acB33d3fA3351Ee43);}/ ** Returns the latest answer. /functiongetChainlinkDataFeedLatestAnswer()publicviewreturns(int){/// prettier-ignore(/*uint80 roundID */,intanswer,/*uint startedAt/,/uint timeStamp/,/uint80 answeredInRound*/)=dataFeed.latestRoundData();returnanswer;}}Open in RemixWhat is Remix?The contract has the following components:

- Theimportline imports an interface namedAggregatorV3Interface. Interfaces define functions without their implementation, which leaves inheriting contracts to define the actual implementation themselves. In this case,AggregatorV3Interfacedefines that all v3 Aggregators have the functionlatestRoundData. You cansee the complete codefor theAggregatorV3Interfaceon GitHub.
- Theconstructor() {}initializes an interface object nameddataFeedthat usesAggregatorV3Interfaceand connects specifically to a proxy aggregator contract that is already deployed at0x1b44F3514812d835EB1BDB0acB33d3fA3351Ee43. The interface allows your contract to run functions on that deployed aggregator contract.
- ThegetChainlinkDataFeedLatestAnswer()function calls yourdataFeedobject and runs thelatestRoundData()function. When you deploy the contract, it initializes thedataFeedobject to point to the aggregator at0x1b44F3514812d835EB1BDB0acB33d3fA3351Ee43, which is the proxy address for the Sepolia BTC / USD data feed. Your contract connects to that address and executes the function. The aggregator connects with several oracle nodes and aggregates the pricing data from those nodes. The response from the aggregator includes several variables, butgetChainlinkDataFeedLatestAnswer()returns only theanswervariable.

Compile, deploy, and run the contract

Configure and fund your wallet

If you have not already configured your MetaMask wallet and funded it with testnet ETH, follow the instructions inDeploy Your First Smart Contractto set that up. You can get testnet ETH at one of the availableSepolia faucets.

Deploy theDataConsumerV3smart contract on the Sepolia testnet.

1. Open the example contractin Remix. Remix opens and shows the contents of the smart contract.

Open the contract in Remix2. Because the code is already written, you can start the compile step. On the left side of Remix, click theSolidity Compilertab to view the compiler settings. 3. Use the default compiler settings. Click theCompile DataConsumerV3.solbutton to compile the contract. Remix automatically detects the correct compiler version depending on thepragmathat you specify in the contract. You can ignore warnings about unused local variables in this example. 4. On theDeploytab, select theInjected Providerenvironment. This contract specifically requires Web3 because it connects with another contract on the blockchain. Running in a JavaScript VM will not work. 5. Because the example contract has several imports, Remix might select another contract to deploy by default. In theContractsection, select theDataConsumerV3contract to make sure that Remix deploys the correct contract. 6.

ClickDeployto deploy the contract to the Sepolia testnet. MetaMask opens and asks you to confirm payment for deploying the contract. Make sure MetaMask is set to the Sepolia network before you accept the transaction. Because these transactions are on the blockchain, they are not reversible. 7. In the MetaMask prompt, clickConfirmto approve the transaction and spend your testnet ETH required to deploy the contract. 8. After a few seconds, the transaction completes and your contract appears under theDeployed Contractslist in Remix. Click the contract dropdown to view its variables and functions. 9. ClickgetChainlinkDataFeedLatestAnswerto show the latest answer from the aggregator contract. In this example, the answer is the latest price, which appears just below the button. The returned answer is an integer, so it is missing its decimal point. You can find the correct number of decimal places for this answer on thePrice Feed addresses page by clicking theShow more detailscheckbox. The answer on the BTC / USD feed uses 8 decimal places, so an answer of3030914000000indicates a BTC / USD price of30309.14. Each feed uses a different number of decimal places for answers.