Using Data Feeds on EVM Chains

The code for reading Data Feeds is the same across all EVM-compatible blockchains and Data Feed types. You choose different types of feeds for different uses, but the request and response format are the same. To read a feed, specify the following variables:

- RPC endpoint URL:This determines which network that your smart contracts will run on. You can use a node provider service or point to your own client . If you are using a Web3 wallet, it is already configured with the RPC endpoints for several networks and the Remix IDE will automatically detect them for you.
- LINK token contract address:The address for the LINK token contract is different for each network. You can find the full list of addresses for all supported networks on the LINK Token Contracts page.
- Feed contract address:This determines which data feed your smart contract will read. Contract addresses are different for each network. You can find the availble contract addresses on the following pages:* Price Feed Addresses
- Proof of Reserve Feed Addresses
- NFT Floor Price Feed Addresses

The examples in this document indicate these variables, but you can modify the examples to work on different networks and read different feeds.

This guide shows example code that reads data feeds using the following languages:

- Onchain consumer contracts:* Solidity
- Vyper
- Offchain reads using Web3 packages:* Javascript with web3.js
- Python with Web3.py
- Golang with go-ethereum

Reading data feeds onchain

These code examples demonstrate how to deploy a consumer contract onchain that reads a data feed and stores the value.

Solidity

To consume price data, your smart contract should reference AggregatorV3Interface , which defines the external functions implemented by 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 Remix What is Remix? ThelatestRoundDatafunction returns five values representing information about the latest price data. See the Data Feeds API Reference for more details.

Vyper

To consume price data, your smart contract should importAggregatorV3Interfacewhich defines the external functions implemented by Data Feeds. You can find it here . You can find aPriceConsumerexample here . Read theapeworx-starter-kit README to learn how to run the example.

Reading data feeds offchain

These code examples demonstrate how to read data feeds directly off chain using Web3 packages for each language.

Javascript

This example uses web3.js to retrieve feed data from the BTC / USD feed on the Sepolia testnet.

web3.jsethers.js/ * **THIS IS EXAMPLE CODE THAT USES HARDCODED VALUES FOR CLARITY. * THIS IS EXAMPLE CODE THAT USES UN-AUDITED CODE. * DO NOT USE THIS CODE IN PRODUCTION. */const Web3=require("web3")// for nodejs onlyconst web3=newWeb3("https://rpc.ankr.com/eth_sepolia")const aggregatorV3InterfaceABI=[{inputs:[],name:"decimals",outputs:[{internalType:"uint8",name:"",type:"uint8"}],stateMutability:"view",type:"function",},{inputs:[],name:"description",outputs:[{internalType:"string",name:"",type:"string"}],stateMutability:"view",type:"function",},{inputs:[{internalType:"uint80",name:"_roundId",type:"uint80"}],name:"getRoundData",outputs:[{internalType:"uint80",name:"roundId",type:"uint80"},{internalType:"int256",name:"answer",type:"int256"},{internalType:"uint256",name:"startedAt",type:"uint256"},{internalType:"uint256",name:"updatedAt",type:"uint256"},{internalType:"uint80",name:"answeredInRound",type:"uint80"},],stateMutability:"view",type:"function",},{inputs:[],name:"latestRoundData",outputs:[{internalType:"uint80",name:"roundId",type:"uint80"},{internalType:"int256",name:"answer",type:"int256"},{internalType:"uint256",name:"startedAt",type:"uint256"},{internalType:"uint256",name:"updatedAt",type:"uint256"},{internalType:"uint80",name:"answeredInRound",type:"uint80"},],stateMutability:"view",type:"function",},{inputs:[],name:"version",outputs:[{internalType:"uint256",name:"",type:"uint256"}],stateMutability:"view",type:"function",},]const addr="0x1b44F3514812d835EB1BDB0acB33d3fA3351Ee43"const priceFeed=newweb3.eth.Contract(aggregatorV3InterfaceABI,addr)priceFeed.methods.latestRoundData().call().then((roundData)=>{// Do something with roundDataconsole.log("Latest Round Data",roundData)})**/ * THIS IS EXAMPLE CODE THAT USES HARDCODED VALUES FOR CLARITY. * THIS IS EXAMPLE CODE THAT USES UN-AUDITED CODE. * DO NOT USE THIS CODE IN PRODUCTION. */const{ethers}=require("ethers")// for nodejs onlyconst provider=newethers.providers.JsonRpcProvider("https://rpc.ankr.com/eth_sepolia")const aggregatorV3InterfaceABI=[{inputs:[],name:"decimals",outputs:[{internalType:"uint8",name:"",type:"uint8"}],stateMutability:"view",type:"function",},{inputs:[],name:"description",outputs:[{internalType:"string",name:"",type:"string"}],stateMutability:"view",type:"function",},{inputs:[{internalType:"uint80",name:"_roundId",type:"uint80"}],name:"getRoundData",outputs:[{internalType:"uint80",name:"roundId",type:"uint80"},{internalType:"int256",name:"answer",type:"int256"},{internalType:"uint256",name:"startedAt",type:"uint256"},{internalType:"uint256",name:"updatedAt",type:"uint256"},{internalType:"uint80",name:"answeredInRound",type:"uint80"},],stateMutability:"view",type:"function",},{inputs:[],name:"latestRoundData",outputs:[{internalType:"uint80",name:"roundId",type:"uint80"},{internalType:"int256",name:"answer",type:"int256"},{internalType:"uint256",name:"startedAt",type:"uint256"},{internalType:"uint256",name:"updatedAt",type:"uint256"},{internalType:"uint80",name:"answeredInRound",type:"uint80"},],stateMutability:"view",type:"function",},{inputs:[],name:"version",outputs:[{internalType:"uint256",name:"",type:"uint256"}],stateMutability:"view",type:"function",},]const addr="0x1b44F3514812d835EB1BDB0acB33d3fA3351Ee43"const priceFeed=newethers.Contract(addr,aggregatorV3InterfaceABI,provider)priceFeed.latestRoundData().then((roundData)=>{// Do something with roundDataconsole.log("Latest Round Data",roundData)})

Latest Price:Latest Price

Python

This example uses Web3.py to retrieve feed data from the BTC / USD feed on the Sepolia testnet.

THIS IS EXAMPLE CODE THAT USES HARDCODED VALUES FOR CLARITY.# THIS IS EXAMPLE CODE THAT USES UN-AUDITED CODE.# DO NOT USE THIS CODE IN PRODUCTION.fromweb3importWeb3

Changethisto use your own RPC URL

web3=Web3(Web3.HTTPProvider('https://rpc.ankr.com/eth_sepolia'))# AggregatorV3Interface ABI abi='[{"inputs":[],"name":"decimals","outputs":[{"internalType":"uint8","name":"","type":"uint8"}],"stateMutability":"view","type":"function"},{"inputs":[],"name":"description","outputs":[{"internalType":"string","name":"","type":"string"}],"stateMutability":"view","type":"function"},{"inputs":[{"internalType":"uint80","name":"_roundId","type":"uint80"}],"name":"getRoundData","outputs":[{"internalType":"uint80","name":"roundId","type":"uint80"},{"internalType":"int256","name":"answer","type":"int256"},{"internalType":"uint256","name":"startedAt","type":"uint256"},{"internalType":"uint256","name":"updatedAt","type":"uint256"},{"internalType":"uint80","name":"answeredInRound","type":"uint80"}],"stateMutability":"view","type":"function"},{"inputs":[],"name":"latestRoundData","outputs":[{"internalType":"uint80","name":"roundId","type":"uint80"},{"internalType":"int256","name":"answer","type":"int256"},{"internalType":"uint256","name":"startedAt","type":"uint256"},{"internalType":"uint256","name":"updatedAt","type":"uint256"},{"internalType":"uint80","name":"answeredInRound","type":"uint80"}],"stateMutability":"view","type":"function"},{"inputs":[],"name":"version","outputs":[{"internalType":"uint256","name":"","type":"uint256"}],"stateMutability":"view","type":"function"}]'# Price Feedaddressaddr='0x1b44F3514812d835EB1BDB0acB33d3fA3351Ee43'# Set upcontractinstancecontract=web3.eth.contract(address=addr,abi=abi)# Make call tolatestRoundData()latestData=contract.functions.latestRoundData().call()print(latestData) Run this Python example

Golang

You can find an example with all the source files here . This example uses go-ethereum to retrieve feed data from the BTC / USD feed on the Sepolia testnet. To learn how to run the example, see the README .

New Feed Registry

You can use theFeed Registry to reference data feed assets by name or currency identifier instead of by pair/proxy address.

Getting a different price denomination

Chainlink Data Feeds can be used in combination to derive denominated price pairs in other currencies.

If you require a denomination other than what is provided, you can use two data feeds to derive the pair that you need. For example, if you needed a BTC / EUR price, you could take the BTC / USD feed and the EUR / USD feed and derive BTC / EUR using division.

Important

If your contracts require Solidity versions that are>=0.6.0 <0.8.0, useOpenZeppelin's SafeMath version 3.4 .

// SPDX-License-Identifier: MITpragmasolidity^0.8.7;import{AggregatorV3Interface}from"@chainlink/contracts/src/v0.8/shared/interfaces/AggregatorV3Interface.sol";/* **Network: Sepolia * Base: BTC/USD * Base Address: 0x1b44F3514812d835EB1BDB0acB33d3fA3351Ee43 * Quote: EUR/USD * Quote Address: 0x1a81afB8146aeFfCFc5E50e8479e826E7D55b910 * Decimals: 8** // * 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. */contractPriceConverter{functiongetDerivedPrice(address_base,address_quote,uint8_decimals)publicviewreturns(int256){require(_decimals>uint8(0)&&_decimals<=uint8(18),"Invalid _decimals");int256decimals=int256(10uint256(_decimals));(,int256basePrice,,,)=AggregatorV3Interface(_base).latestRoundData();uint8baseDecimals=AggregatorV3Interface(_base).decimals();basePrice=scalePrice(basePrice,baseDecimals,_decimals);(,int256quotePrice,,,)=AggregatorV3Interface(_quote).latestRoundData();uint8quoteDecimals=AggregatorV3Interface(_quote).decimals();quotePrice=scalePrice(quotePrice,quoteDecimals,_decimals);ret{if(_priceDecimals<_decimals){return_priceint256(10uint256(_decimals-_priceDecimals));}elseif(_priceDecimals>_decimals){return_price/int256(10*uint256(_priceDecimals-_decimals));}return_price;}}
Open in Remix What is Remix?

More aggregator functions

Getting the latest price is not the only data that aggregators can retrieve. You can also retrieve historical price data. To learn more, see thHistorical Price Data page.

To understand different use cases for Chainlink Price Feeds, refer toOther Tutorials .