Data Feeds API Reference

When you use data feeds, retrieve the feeds through theAggregatorV3Interfaceand the proxy address. Optionally, you can call variables and functions in theAccessControlledOffchainAggregatorcontract to get information about the aggregator behind the proxy.

AggregatorV3Interface

Import this interface to your contract and use it to run functions in the proxy contract. Create the interface object by pointing to the proxy address. For example, on Sepolia you could create the interface object in the constructor of your contract using the following example:

/ * Network: Sepolia * Data Feed: ETH/USD * Address: 0x694AA1769357215DE4FAC081bf1f309aDC325306* /constructor() {priceFeed=AggregatorV3Interface(0x694AA1769357215DE4FAC081bf1f309aDC325306);} To see examples for how to use this interface, read theUsing Data Feeds guide.

You can see the code for theAggregatorV3Interfacecontract on GitHub.

Functions in AggregatorV3Interface

NameDescriptiondecimalsThe number of decimals in the responsedescriptionThe description of the aggregator that the proxy points togetRoundDataGet data from a specific round.latestRoundDataGet data from the latest roundversionThe version representing the type of aggregator the proxy points to.

decimals

Get the number of decimals present in the response value.

functiondecimals()externalviewreturns(uint8); * RETURN: The number of decimals.

description

Get the description of the underlying aggregator that the proxy points to.

functiondescription()externalviewreturns(stringmemory); * RETURN: The description of the underlying aggregator.

getRoundData

Get data about a specific round, using theroundId.

functiongetRoundData(uint80_roundId)externalviewreturns(uint80roundId,int256answer,uint256startedAt,uint256updatedAt,uint80answeredInRound); Parameters:

- _roundId: The round ID

Return values:

- roundId: The round ID
- answer: The answer for this round
- startedAt: Timestamp of when the round started
- updatedAt: Timestamp of when the round was updated
- answeredInRound:Deprecated - Previously used when answers could take multiple rounds to be computed

latestRoundData

Get the data from the latest round.

functionlatestRoundData()externalviewreturns(uint80roundId,int256answer,uint256startedAt,uint256updatedAt,uint80answeredInRound) Return values:

- roundId: The round ID.
- answer: The data that this specific feed provides. Depending on the feed you selected, this answer provides asset prices, reserves, NFT floor prices, and other types of data.
- startedAt: Timestamp of when the round started.
- updatedAt: Timestamp of when the round was updated.
- answeredInRound:Deprecated - Previously used when answers could take multiple rounds to be computed

version

The version representing the type of aggregator the proxy points to.

functionversion()externalviewreturns(uint256) * RETURN: The version number.

AccessControlledOffchainAggregator

This is the contract for the aggregator. You can call functions on the aggregator directly, but it is a best practice to use thAggregatorV3Interface to run functions on the proxy instead so that changes to the aggregator do not affect your application. Read the aggregator contract only if you need functions that are not available in the proxy.

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 thlibocr repository on GitHub.

Variables and functions in AccessControlledOffchainAggregator

This contract importsOffchainAggregatorandSimpleReadAccessController, which also include their own imports. The variables and functions lists include the publicly accessible items from these imported contracts.

A simple way to read the variables or functions is to get the ABI from a blockchain explorer and point the ABI to the aggregator address. To do this in Remix, follow thUsing the ABI with AtAddress guide in the Remix documentation. As an example, you can find the ABI for the BTC/USD aggregator by viewing thecontract code in Etherscan .

Variables:

NameDescriptionLINKThe address for the LINK token contract on a specific network.billingAccessControllerThe address for the billingAccessController, which limits access to thbilling configuration for the aggregator.checkEnabledA boolean that indicates if access is limited to addresses on the internal access list.maxAnswerThis value is no longer used on most Data Feeds. Evaluate if your use case for Data Feeds requires a custom circuit breaker and implement it to meet the needs of your application. See theRisk Mitigation page for more information.minAnswerThis value is no longer used on most Data Feeds. Evaluate if your use case for Data Feeds requires a custom circuit breaker and implement it to meet the needs of your application. See theRisk Mitigation page for more information.ownerThe address that owns this aggregator contract. This controls which address can execute specific functions. Functions:

NameDescriptiondecimalsReturn the number of digits of precision for the stored answer. Answers are stored in fixed-point formatdescriptionReturn a description for this data feed. This is different depending on which feed you select.getAnswerDeprecated - Do not use this functiongetBillingRetrieve the current billing configuration.getRoundDataGet the full information for a specific aggregator round including the answer and update timestamps. Use this to get the full historical data for a round.getTimestampDeprecated - Do not use this functionhasAccessCheck if an address has internal access.latestAnswerDeprecated - Do not use this functionlatestConfigDetailsReturn information about the current offchain reporting protocol configurationlatestRoundDeprecated - Do not use this function.latestRoundDataGet the full information for the most recent round including the answer and update timestampslatestTimestampDeprecated - Do not use this function.latestTransmissionDetailsGet information about the most recent answerlinkAvailableForPaymentGet the amount of LINK on this contract that is available to make payments to oracles. This value can be negative if there are outstanding payment obligations.oracleObservationCountReturns the number of observations that oracle is due to be reimbursed forowedPaymentReturns how much LINK an oracle is owed for its observations.requesterAccessControllerReturns the address for the access controller contract.transmittersThe oracle addresses that can report answers to this aggregator.typeAndVersionReturns the aggregator type and version. Many aggregators areAccessControlledOffchainAggregator 3.0.0, but there are other variants in production. The version is for the type of aggregator, and different from the contractversion.validatorConfigReturns the address and the gas limit for the validator contractversionReturns the contract version. This is different from thetypeAndVersionfor the aggregator.

decimals

Return the number of digits of precision for the stored answer. Answers are stored in fixed-point format.

functiondecimals()externalviewreturns(uint8decimalPlaces);

description

Return a description for this data feed. Usually this is an asset pair for a price feed.

functiondescription()publicviewoverride checkAccessreturns(stringmemory){returnsuper.description();}

getAnswer

This function is deprecated. Do not use this function.

getBilling

Retrieve the current billing configuration.

functiongetBilling()externalviewreturns(uint32maximumGasPrice,uint32reasonableGasPrice,uint32microLinkPerEth,uint32linkGweiPerObservation,uint32linkGweiPerTransmission)
{Billingmemorybilling=s_billing;return(billing.maximumGasPrice,billing.reasonableGasPrice,billing.microLinkPerEth,billing.linkGweiPerObservation,billing.linkGweiPerTransmission);}

getRoundData

Get the full information for a specific aggregator round including the answer and update timestamps. Use this to get the full historical data for a round.

functiongetRoundData(uint80_roundId)publicviewoverride checkAccessreturns(uint80roundId,int256answer,uint256startedAt,uint256updatedAt,uint80answeredInRound)
{returnsuper.getRoundData(_roundId);}

getTimestamp

This function is deprecated. Do not use this function.

hasAccess

Check if an address has internal access.

functionhasAccess(address_user,bytesmemory_calldata)publicviewvirtual overridereturns(bool){returnsuper.hasAccess(_user,_calldata)||_user==tx.origin;}

latestAnswer

This function is deprecated. Do not use this function.

latestConfigDetails

Return information about the current offchain reporting protocol configuration.

functionlatestConfigDetails()externalviewreturns(uint32configCount,uint32blockNumber,bytes16configDigest){return(s_configCount,s_latestConfigBlockNumber,s_hotVars.latestConfigDigest);}

latestRound

This function is deprecated. Do not use this function.

latestRoundData

Get the full information for the most recent round including the answer and update timestamps.

functionlatestRoundData()publicviewoverride checkAccessreturns(uint80roundId,int256answer,uint256startedAt,uint256updatedAt,uint80answeredInRound){returnsuper.latestRoundData();}

latestTimestamp

This function is deprecated. Do not use this function.

latestTransmissionDetails

Get information about the most recent answer.

functionlatestTransmissionDetails()externalviewreturns(bytes16configDigest,uint32epoch,uint8round,int192latestAnswer,uint64latestTimestamp){require(msg.sender==tx.origin,"Only callable by
EOA");return(s_hotVars.latestConfigDigest,uint32(s_hotVars.latestEpochAndRound>>8),uint8(s_hotVars.latestEpochAndRound),s_transmissions[s_hotVars.latestAggregatorRoundId].answer,s_transmis

linkAvailableForPayment

Get the amount of LINK on this contract that is available to make payments to oracles. This value can be negative if there are outstanding payment obligations.

functionlinkAvailableForPayment()externalviewreturns(int256availableBalance){// there are at most one billion LINK, so this cast is safeint256balance=int256(LINK.balanceOf(address(this)));// according to the argument in the definition of totalLINKDue,// totalLINKDue is never greater than 2**172, so this cast is safeint256due=int256(totalLINKDue());// safe from overflow according to above sizesreturnint256(balance)-int256(due);}

oracleObservationCount

Returns the number of observations that oracle is due to be reimbursed for.

functionoracleObservationCount(address_signerOrTransmitter)externalviewreturns(uint16){Oraclememoryoracle=s_oracles[_signerOrTransmitter];if(oracle.role==Role.Unset)
{return0;}returns_oracleObservationsCounts[oracle.index]-1;}

owedPayment

Returns how much LINK an oracle is owed for its observations.

functionowedPayment(address_transmitter)publicviewreturns(uint256){Oraclememoryoracle=s_oracles[_transmitter];if(oracle.role==Role.Unset)
{return0;}Billingmemorybilling=s_billing;uint256linkWeiAmount=uint256(s_oracleObservationsCounts[oracle.index]-
1)*uint256(billing.linkGweiPerObservation)*(1gwei);linkWeiAmount+=s_gasReimbursementsLinkWei[oracle.index]-1;returnlinkWeiAmount;}

requesterAccessController

Returns the address for the access controller contract.

functionrequesterAccessController()externalviewreturns(AccessControllerInterface){returns_requesterAccessController;}

transmitters

The oracle addresses that can report answers to this aggregator.

functiontransmitters()externalviewreturns(address[]memory){returns_transmitters;}

typeAndVersion

Returns the aggregator type and version. Many aggregators areAccessControlledOffchainAggregator 2.0.0, but there are other variants in production. The version is for the type of aggregator, and different from the contractversion.

functiontypeAndVersion()externalpurevirtual overridereturns(stringmemory){return"AccessControlledOffchainAggregator 2.0.0";}

validatorConfig

Returns the address and the gas limit for the validator contract.

functionvalidatorConfig()externalviewreturns(AggregatorValidatorInterface validator,uint32gasLimit){ValidatorConfigmemoryvc=s_validatorConfig;return(vc.validator,vc.gasLimit);}

version

Returns the contract version. This is different from thetypeAndVersionfor the aggregator.

functionversion()externalviewreturns(uint256);