ValidatorsExitBusOracle

- Source code
- Deployed contract
- InheritsBaseOracle

info It's advised to readWhat is Lido Oracle mechanism before

What is ValidatorsExitBusOracle

A contract that implements an on-chain "source of truth" message bus between the protocol's off-chain oracle and off-chain observers, with the main goal of delivering validator exit requests to the Lido-participating node operators.

A report calculation consists of 4 key steps:

1. Calculate withdrawals amount to cover with ether.
2. Calculate ether rewards prediction per epoch.
3. Calculate withdrawal epoch for next validator eligible for exit to cover withdrawal requests if needed
4. Prepare validators exit order queue
5. Go through the queue until the exited validators' balances cover all withdrawal requests (considering the predicated final exited balance of each validator).

note Placed exit requests viaValidatorsExitBusOracle should be processed timely according to the ratifiedLido on Ethereum Validator Exits Policy V1.0 . Access to lever methods is restricted using the functionality of theAccessControlEnumerable contract and a bunch ofgranular roles .

Report cycle

The oracle work is delineated by equal time periods called frames. In normal operation, oracles finalize a report in each frame (the frame duration is 75 Ethereum Consensus Layer epochs, each frame starts at ~04:00, ~12:00, ~20:00 UTC). Each frame has a reference slot and processing deadline. Report data is gathered by looking at the world state (both Ethereum Execution and Consensus Layers) at the moment of the frame's reference slot (including any state changes made in that slot), and must be processed before the frame's processing deadline.

Reference slot for each frame is set to the last slot of the epoch preceding the frame's first epoch. The processing deadline is set to the last slot of the last epoch of the frame.

It's worth noting that frame lengthcan be changed . And if oracle report is delayed it does not extend the report period, unless it's missed. In this case, the next report will have the report period increased.

The frame includes these stages:

- Waiting
-
    - oracle starts as adaemon
- and wakes up every 12 seconds (by default) in order to find the last finalized slot, trying to collate with it with the expected reference slot;
- Data collection
- : oracles monitor the state of both the execution and consensus layers and collect the data for the successfully arrived finalized reference slot;
- Hash consensus
- : oracles analyze the report data, compile the report and submit its hash to theHashConsensus
- smart contract;
- Core update report
- : once thequorum
- of hashes is reached, meaning more than half of the oracles submitted the same hash (i.e., 5 of 9 oracle committee members at the moment of writing), one of the oracles chosen in turn submits the actual report to theValidatorsExitBusOracle
- contract, which triggers a chain of theValidatorExitRequest
- events containing details about the next validators to be ejected (to initiate a voluntary exit from the Ethereum Consensus Layer side).

Report data

The functionsubmitReportData() accepts the followingReportData structure.

struct

ReportData

{ uint256 consensusVersion ; uint256 refSlot ; uint256 requestsCount ; uint256 dataFormat ; bytes data ; } Oracle consensus info

- consensusVersion
- — Version of the oracle consensus rules. A current version expected by the oracle can be obtained by callinggetConsensusVersion()
- .
- refSlot
- — Reference slot for which the report was calculated. The state being reported must include all state changes resulting from the all blocks up to this reference slot (inclusive). The epoch containing the slot must be finalized prior to calculating the report.

Requests data

- requestsCount
- — Total number of validator exit requests in this report. Must not be greater
- than the limit enforced byOracleReportSanityChecker.checkExitBusOracleReport
- .
- dataFormat
- — Format of the validator exit requests data. Currently, only theDATA_FORMAT_LIST=1
- value is supported.
- data
- — Validator exit requests data. Can differ based on the data format, see the constant defining a specific data formathere
- for more info.

Constants

DATA_FORMAT_LIST()

The list format of the validator exit requests data.

note Each validator exit request is described by the following 64-byte array:

MSB <---------- LSB | 3 bytes | 5 bytes | 8 bytes | 48 bytes | | moduleId | nodeOpId | validatorIndex | validatorPubkey | All requests are tightly packed into a byte array where requests follow one another without any separator or padding, and passed to thedata field of the report structure.

Requests must be sorted in the ascending order by the following compound key:(moduleId, nodeOpId, validatorIndex) . uint256

public

constant DATA_FORMAT_LIST =

1

SECONDS_PER_SLOT()

Seehttps://ethereum.org/en/developers/docs/blocks/#block-time

note always returns 12 seconds due tothe Merge uint256

public immutable SECONDS_PER_SLOT

GENESIS_TIME()

Seehttps://blog.ethereum.org/2020/11/27/eth2-quick-update-no-21

note always returns 1606824023 (December 1, 2020, 12:00:23pm UTC) onMainnet uint256

public immutable GENESIS_TIME

PAUSE_INFINITELY()

Special value for the infinite pause. SeepauseFor andpauseUntil .

uint256

public

constant PAUSE_INFINITELY =

type (uint256) . max

ProcessingState

struct

ProcessingState

{ uint256 currentFrameRefSlot ; uint256 processingDeadlineTime ; bytes32 dataHash ; bool dataSubmitted ; uint256 dataFormat ; uint256 requestsCount ; uint256 requestsSubmitted ; } * currentFrameRefSlot * — Reference slot for the current reporting frame. * processingDeadlineTime * — The last time at which a report data can be submitted for the current reporting frame. * dataHash * — Hash of the report data. Zero bytes if consensus on the hash hasn't been reached yet for the current reporting frame. * dataSubmitted * — Whether any report data for the for the current reporting frame has been already submitted. * dataFormat * — Format of the report data for the current reporting frame. * requestsCount * — Total number of validator exit requests for the current reporting frame. * requestsSubmitted * — How many validator exit requests are already submitted for the current reporting frame.

View methods

getTotalRequestsProcessed()

Returns the total number of validator exit requests ever processed across all received reports.

function

getTotalRequestsProcessed ( )

external

view

returns

(uint256)

getLastRequestedValidatorIndices()

Returns the latest validator indices that were requested to exit for the givennodeOpIds in the givenmoduleId . For node operators that were never requested to exit any validator, index is set to-1 .

function

getLastRequestedValidatorIndices (uint256 moduleId ,

uint256 [ ]

calldata nodeOpIds) external

view

returns

(int256 [ ]

memory)

Parameters

Name Type Description moduleId uint256 ID of the staking module. nodeOpIds uint256 IDs of the staking module's node operators.

Reverts

- Reverts withArgumentOutOfBounds()
- ifmoduleId > UINT24_MAX
- Reverts withArgumentOutOfBounds()

- ifnodeOpId > UINT40_MAX

getProcessingState()

Returns data processing state for the current reporting frame. See the docs for theProcessingState struct.

function

getProcessingState ( )

external

view

returns

(ProcessingState memory result)

getConsensusContract()

Returns the address of theHashConsensus contract instance used byValidatorsExitBusOracle .

function

getConsensusContract ( )

external

view

returns

(address)

getConsensusReport()

Returns the last consensus report hash and metadata.

function

getConsensusReport ( )

external

view

returns

(bytes32 hash , uint256 refSlot , uint256 processingDeadlineTime , bool processingStarted)

getConsensusVersion()

Returns the current consensus version expected by the oracle contract.

note Consensus version must change every time consensus rules change, meaning that an oracle looking at the same reference slot would calculate a different hash. function

getConsensusVersion ( )

external

view

returns

(uint256)

getContractVersion()

Returns the current contract version.

function

getContractVersion ( )

public

view

returns

(uint256)

getLastProcessingRefSlot()

Returns the last reference slot for which processing of the report was started.

function

getLastProcessingRefSlot ( )

external

view

returns

(uint256)

Returns

Name Type Description hash bytes32 The last reported hash refSlot uint256 The frame's reference slot: if the data the consensus is being reached upon includes or depends on any onchain state, this state should be queried at the reference slot. If the slot contains a block, the state should include all changes from that block. processingDeadlineTime uint256 Timestamp of the last slot at which a report can be reported and processed processingStarted bool Has the processing of the report been started or not

getResumeSinceTimestamp()

Returns one of thetimestamp values:

- PAUSE_INFINITELY
- if paused permanently (i.e., with no expiration timestamp)
- a first second when get contract get resumed if paused for specific duration (iftimestamp ≥ block.timestamp
-)
- some timestamp in past if not paused (iftimestamp < block.timestamp
-)

function

getResumeSinceTimestamp ( )

external

view

returns

(uint256 timestamp)

isPaused()

Returns whether the contract is paused or not at the moment.

function

isPaused ( )

public

view

returns

(bool)

Methods

submitReportData()

Submits report data for processing.

function

submitReportData (ReportData calldata data ,

uint256 contractVersion) external whenResumed

Parameters

Name Type Description data ReportData The report data. SeeReportData for details. contractVersion uint256 Expected version of the oracle contract.

Reverts

- Reverts withSenderNotAllowed()
- if the caller doesn't have aSUBMIT_DATA_ROLE
- role and is not a member of the oracle committee.
- Reverts withUnexpectedContractVersion(expectedVersion, version)
- if the provided contract version differs from the current one.
- Reverts withUnexpectedConsensusVersion(expectedConsensusVersion, consensusVersion)
- if the provided consensus version differs from the expected one.
- Reverts withUnexpectedRefSlot(report.refSlot, refSlot)
- if the provided reference slot differs from the current consensus frame's one.
- Reverts withUnexpectedDataHash(report.hash, hash)
- if akeccak256
- hash of the ABI-encoded data differs from the last hash.
- Reverts withNoConsensusReportToProcess()
- if the report hash data is0
- .
- Reverts withRefSlotAlreadyProcessing()
- if the report reference slot is equal to the previous processing reference slot.
- Reverts withInvalidRequestsData()
- ifmoduleId
- in the provided data is0
- Reverts withInvalidRequestsDataLength()
- if the provided data is packed incorrectly
- Reverts withUnexpectedRequestsDataLength()
- if the length of the provided packed data is not equaldata.requestsCount
- Reverts withInvalidRequestsDataSortOrder
- when the provided data is not sorted
- Reverts withNodeOpValidatorIndexMustIncrease(uint256 moduleId, uint256 nodeOpId, uint256 prevRequestedValidatorIndex, uint256 requestedValidatorIndex)
- ifrequested validator index <= last requested index
- from the same module

pauseFor()

Pause accepting the reports data and forming new validator exit requests for the provided duration in seconds.

function

pauseFor (uint256 _duration)

external

Parameters

Name Type Description _duration uint256 pause duration, seconds (usePAUSE_INFINITELY for unlimited)

Reverts

- Reverts withResumedExpected()

- if contract is already paused
- Reverts withAccessControl:...
- reason if sender has noPAUSE_ROLE
- Reverts withZeroPauseDuration()
- if zero duration is passed

pauseUntil()

Pause accepting the reports data and forming new validator exit requests till the given timestamp (inclusive).

function

pauseUntil (uint256 _pauseUntilInclusive)

external

Parameters

Name Type Description _pauseUntilInclusive uint256 the last second to pause until (inclusive)

Reverts

- Reverts withResumeSinceInPast()
- if the provided timestamp is in the past
- Reverts withAccessControl:...
- reason if the sender has noPAUSE_ROLE
- Reverts withResumedExpected()
- if the contract is already paused

resume()

Resume accepting the reports data and forming new validator exit requests.

function

resume ( )

external

Reverts

- Reverts withPausedExpected()
- if contract is already resumed (i.e., not paused)
- Reverts withAccessControl:...
- reason if the sender has noRESUME_ROLE

Permissions

SUBMIT_DATA_ROLE()

An ACL role granting the permission to submit the data for a committee report.

bytes32

public

constant SUBMIT_DATA_ROLE =

keccak256 ("SUBMIT_DATA_ROLE")

PAUSE_ROLE()

An ACL role granting the permission to pause accepting the reports data and forming new validator exit requests.

bytes32

public

constant PAUSE_ROLE =

keccak256 ("PAUSE_ROLE")

RESUME_ROLE()

An ACL role granting the permission to resume accepting the reports data and forming new validator exit requests.

bytes32

public

constant RESUME_ROLE =

keccak256 ("RESUME_ROLE")

MANAGE_CONSENSUS_CONTRACT_ROLE()

An ACL role granting the permission to set the consensus contract address by callingsetConsensusContract .

bytes32

public

constant MANAGE_CONSENSUS_CONTRACT_ROLE =

keccak256 ("MANAGE_CONSENSUS_CONTRACT_ROLE") ;

MANAGE_CONSENSUS_VERSION_ROLE()

An ACL role granting the permission to set the consensus version by callingsetConsensusVersion .

bytes32

public

constant MANAGE_CONSENSUS_VERSION_ROLE =

keccak256 ("MANAGE_CONSENSUS_VERSION_ROLE") ;

Events

ValidatorExitRequest()

Emits when the new report data submitted for processing.

event

ValidatorExitRequest (uint256

indexed stakingModuleId , uint256

indexed nodeOperatorId , uint256

indexed validatorIndex , bytes validatorPubkey , uint256 timestamp)

WarnDataIncompleteProcessing()

Emits on attempt of new data submission having not all of the items processed yet.

event

WarnDataIncompleteProcessing (uint256

indexed refSlot , uint256 requestsProcessed , uint256 requestsCount)

Paused()

Emits when the contract is paused either by thepauseFor orpauseUntil calls.

event

Paused (uint256 duration)

Resumed()

Emits when the contract is resumed by theresume call.

event

Resumed ( ) Edit this page Previous AccountingOracle Next HashConsensus