Validators Exit Bus

info It's advised to read What is Lido Oracle mechanism before Validators Exit Bus is an oracle that ejects Lido validators when the protocol requires additional funds to process user withdrawals.

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 via ValidatorsExitBusOracle should be processed timely according to the ratified Lido on Ethereum Validator Exits Policy V1.0 .

See also the provided penalties spec.

Next validator to exit algorithm

The algorithm for the validators exiting is based on the algorithm described on the research forum.

The algorithm is supposed to correct the future number of validators for each Node Operator. Suppose the validators and deposits in-flight of one of the Node Operator are represented in the following form, where validators are sorted by their indexes:

The algorithm assumes that the oldest validators are exited first. Therefore, previously requested validators can be separated to exit by knowing the index of the last requested.

Worth noting, each validator has a status. Some validators may be slashed or be exited without an request from the protocol:

Among all validators the projected ones are the point of interest. They include all active validators and in-flight deposits, but exclude validators whose exit_epoch != FAR_FUTURE_EPOCH and those validators that were requested to exit.

A few hours later it might look like the following:

Note that th described algorithm is looking for a validator to exit only among those that can be exited, while using the projected number of validators, which includes non-existent yet validators. It's only weights, so there is no misconception here.

The final exit order predicate sequence:

1. Validator whose operator with the lowest number of delayed validators
2. Validator whose operator with the highest number of targeted validators to exit
3. Validator whose operator with the highest stake weight
4. Validator whose operator with the highest number of validators
5. Validator with the lowest index

Get information to prepare ordered queue

In order to prepare a queue of validators to exit, the following actions and considerations involved:

- the maximum number of validators that can be requested to exit in one report;
- operator network penetration percent - only if the operator's share is greater than 1%;
- 'exitable' Lido validators;
- fetch node operators stats to sort exitable validators;
- total predictable validators count;
- last requested validators indices.

Report limits

- maxValidatorExitRequestsPerReport
-
    - max number of exit requests allowed in report to ValidatorsExitBusOracle
- from OracleReportSanityChecker.getOracleReportLimits()
- .

- VALIDATOR_DELAYED_TIMEOUT_IN_SLOTS
-
  - A parameter from fromOracleDaemonConfig
- contract used to calculate validators going to exit.
- NODE_OPERATOR_NETWORK_PENETRATION_THRESHOLD_BP
-
  -
    - A parameter fromOracleDaemonConfig
- that is taken into account when determining the penetration of the operator into the network.

Get exitable validators

A validator is 'exitable' if two conditions are strictly have NOT met:

- validator.exit_epoch != FAR_FUTURE_EPOCH
- and
- validator.index <= last_requested_to_exit_index
- .

Node operator stats

Statistics for each node operator, which are needed for sorting their validators in exit order:

- validators count that are not yet in CL
- validators that are in CL and are not yet requested to exit and not on exit
- validators that are in CL and requested to exit but not on exit and not requested to exit recently
- target validators count
- checks whether the target limit flag is enabled

NB: A validator can not be considered as delayed if it was requested to exit in lastVALIDATOR_DELAYED_TIMEOUT_IN_SLOTS slots

Last requested validators indices

TheValidatorsExitBusOracle contract stores the index of the last validator that was requested to exit. Since validators are requested in strict order from the lowestvalidatorIndex to the highest, the indexes help find all the previously requested validators without fetching all events.

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

ValidatorsExitBusOracle.getLastRequestedValidatorIndices(uint256 moduleId, uint256[] nodeOpIds): int256[]

State collection

To find the next validators to exit, Validators Exit Bus Oracle collects the following state from both Ethereum Consensus and Execution layers.

- FromOracleDaemonConfig
- contract:* PREDICTION_DURATION_IN_SLOTS
-
  - VALIDATOR_DELAYED_TIMEOUT_IN_SLOTS
- FromWithdrawal Queue
- :* Get total unfinalized withdrawal request amount
- FromLido
- contract:* Recent postCLBalance/preCLBalance and withdrawals from Execution Layer Rewards and Withdrawal vaults via events
- From Consensus Layer node:* All validators and their states on the reference slot
- FromStaking Router
- :* Public keys of all Lido validators
-
  - Indices of the last requested validator to exit for each Node Operator
-
  - Validator keys statistics for each Node Operator
- From Oracle contract:* Maximum number of exit requests for the current frame
-
  - Recently requested via Exit Bus public keys to exit

Fetching data

Get uncovered withdrawal requests amount of stETH

Collects the amount of stETH in the queue yet to be finalized fromWithdrawalQueue.unfinalizedStETH()

Calculate average rewards speed per epoch

FetchesETHDistributed andTokenRebased events from theLido contract and calculate average rewards amount per epoch. The rewards prediction period config fetches from theOracleDaemonConfig contract.

To get events in past, addressing the cases where there can be slots with missed block, the next scheme is introduced:

- Get fromOracleDaemonConfig
- contractPREDICTION_DURATION_IN_SLOTS
- value
- GetTokenRebased
- events from Lido
- GetETHDistributed
- events from Lido
- Group that events by transaction hash
- Collect from events:* total_rewards
-
    - aspostCLBalance + withdrawalsWithdrawn - preCLBalance executionLayerRewardsWithdrawn
-
    - time_spent
-
    - as sum of each eventtimeElapsed
- calculaterewards_speed_per_epoch
- asmax(total_rewards * chain_configs.seconds_per_slot * chain_configs.slots_per_epoch // time_spent, 0)

Calculate epochs to sweep

Average sweep prediction

Predicts the average epochs of the sweep cycle. In the specget expected withdrawals ,process withdrawals

source

Withdrawable validators

- Check ifvalidator
- has the 0x01 prefixed "eth1" withdrawal credentials, and
- Check ifvalidator
- is partially withdrawable, or
- Check ifvalidator
- is fully withdrawable

source

Predict available ether before next withdrawn

In order to estimate the amount is needed to fully cover the non-finalized withdraw requests, the following values are calculated

- Future rewards
- Future withdrawals amount
- Total available balance
- Validators to eject cummulative amount
- Going to withdrawn balance

To calculatefuture rewards , it's needed topredict an epoch when all validators in queue andvalidators_to_eject will be withdrawn:

1. Calculate latest exit epoch number and amount of validators that are exiting in this epoch
2. If queue is empty - exit epoch will be calculated ascurrent epoch + MAX_SEED_LOOK AHEAD + 1
3. .MAX_SEED_LOOKAHEAD
4. constant needs to mitigate some attacks, more detailshere
5. Calculatechurn limit
6.
    - like a rate-limit on changes to the validator set. Minimum is 4 validators per epoch. And recalculates

eachCHURN_LIMIT_QUOTIENT = 2**16
7. . For example when active validators reaches up to 327,680 amount,churn limit
8. rises to 5,spec
9. Calculate slots capacity for exit:

remain_exits_capacity_for_epoch=churn_limit - (amount of validators that are exiting in this epoch) 1. Calculate epoch to exit allvalidators_to_eject_count 2. :

epochs_required_to_exit_validators = (validators_to_eject_count - remain_exits_capacity_for_epoch) // churn_limit + 1 1. So the predictable withdrawable epoch:

withdrawal_epoch=max_exit_epoch_number + epochs_required_to_exit_validators + MIN_VALIDATOR_WITHDRAWABILITY_DELAY) MIN_VALIDATOR_WITHDRAWABILITY_DELAYhere

So now we can calculate what amount (and validators count) is needed to fully cover amount of non-finalized WithdrawQueue requests.

Calculate expected balance to withdraw

Future rewards

future_rewards = (withdrawal_epoch + epochs_to_sweep - blockstamp.ref_epoch) * rewards_speed_per_epoch

Future withdrawals amount

Get total balance from validators which can be fully withdrawn.

Total available balance

Fetch total balance as sum from:

- Lido.getBufferedEther()
- +
- Balance fromelRewardsVault
- +
- Balance fromwithdrawalVault

Validators to eject cummulative amount

Get balance from next validator in exit queue.

Validators going to exit

Fetches recently emittedValidatorExitRequest events fromValidatorsExitBusOracle contract and extract pubkeys from them. The delayed timeout config fetches from theOracleDaemonConfig contract.

Validators requested to exit, but didn't send exit message. In case:

- Activation epoch is not old enough to initiate exit
- Node operator had not enough time to send exit message (VALIDATOR_DELAYED_TIMEOUT_IN_SLOTS)

To get validators, oracle calculates:

- lido_validators_by_operator
-
    - Fetches all used Lido keys fromKeys API
-
    - Fetches all validators at the reference slot and merge them with keys
- ejected_indexes
-
    - get operators with last exited validator indexes from for all staking_modules and node operators viaValidatorsExitBusOracle.getLastRequestedValidatorIndices(module_id, uint256[] nodeOpIds)
- recent_pubkeys
-
    - get last requested to exit pubkeys fromValidatorExitRequest
- event

For eachlido_validators_by_operator oracle tries to findnon exited validators , so:

- if notvalidator_asked_to_exit
- -> return False
- ifis_on_exit
- -> return false
- ifvalidator_recently_asked_to_exit
- -> returnTrue
- if notvalidator_eligible_to_exit
- -> returnTrue
- otherwise return False

Oracle calculatesgoing_to_withdraw_balance for allnon exited validators

Compare expected_balance vs to_withdrawn_balance

Expected balance is:

expected_balance = (future_withdrawals + # Validators that have withdrawal_epoch future_rewards + # Rewards we get until last validator in validators_to_eject will be withdrawn total_available_balance + # Current EL balance (el vault, wc vault, buffered eth) validator_to_eject_balance_sum + # Validators that we expected to be ejected (requested to exit, not delayed) going_to_withdraw_balance # validators_to_eject balance) First of all, it's checked without exiting the validator, whether the protocol already has enough available ether to cover withdrawal requests in the queue. If yes, then it's not reasonable to exit validators.

If there is not enough, one more validator is considered to be exited and the expected balance gets calculated again. The process continues until the expected balance becomes greater than or equal to the unfinalized withdrawal requests amount.

Helpful links

- Lido Oracle source code Edit this page Previous Accounting oracle Next Validator exits and penalties