NodeOperatorsRegistry

- Source Code
- Deployed Contract

TheNodeOperatorsRegistry contract acts as a registry of Node Operators selected by the Lido DAO. Since Lido V2 upgrade NodeOperatorsRegistry contract became a module ofStakingRouter and got the second nameCurated staking module as part of the general Lido staking infrastructure. As a staking module,NodeOperatorsRegistry implementsStakingModule interface .

NodeOperatorsRegistry keeps track of various Node Operators data, in particular limits of the allowed stake, reward addresses, penalty information, public keys of the Node Operators' validators. It defines order in which the Node Operators get the ether deposited and reward distribution between the node operators.

A curated node operator is obliged by the Lido DAO to exit its validators timely if requested by the Lido protocol. The exit request is formed on-chain by means ofValidatorsExitBusOracle contract. If a NO doesn't fulfil the request timely, it might get penalized. The penalized status is assigned automatically by the Lido protocol. The penalized NO do not get new ether for new deposits and also receives half of its rewards till the penalty is cleared. The other half of the NO rewards gets distributed between all stETH holders (technically, it gets burned). To get the penalty cleared, the NO must exit the stuck validators or refund the corresponding ether amount and waitgetStuckPenaltyDelay() seconds after that.

The Lido DAO can also:

- set target limit count as the number of validators for the NO. If the current active number of validators is below the value, the excess ones will be requested to exit in a prioritized manner when required tofinalize withdrawal requests
- . Allocation of deposits above the target value is prohibited.
- deactivate misbehaving operators bydeactivateNodeOperator()
- . A deactivated node operator do not get rewards and new deposits.

Glossary

note In the context of these terms "signing key", "key", "validator key", "validator" might be used interchangeably. signing key . BLS12-381 public key that will be used by the protocol for making Beacon deposits torun a validator

vetted (signing key). Approved by the Lido DAO for receiving ether for deposit.

submitted (signing key). Added to the node operators registry.

depositable (signing key). Suitable for new deposits.

deposited (signing key). Ever received deposit.

unused (signing key). Submitted but not deposited yet.

exited (signing key). A validator that got into "Exited" state: either byvoluntary exit or as a result of slashing.This doc might be useful regarding the validators lifecycle.

used (active) (signing key). Deposited but not yet exited.

stuck (validator). Not exited in proper time after an exit request fromValidatorsExitBusOracle by Lido protocol.

refunded (stuck validator). Compensated by the NO for being stuck. For more information on handling of NO misbehavior seeLido on Ethereum Block Proposer Rewards Policy v1.0.

Node operator parameters

For each NO the contract keeps a record of at least these values:

- active: bool
- active/inactive status of the NO. An active NO gets rewards and new deposits according to its staking limit. New node operators are added in active state.
- name: string
- human-readable name of the NO
- rewardAddress: address
- where to send stETH rewards (part of the protocol fee)
- totalVettedValidators: uint64
- Max number of validator keys approved for deposit by the DAO so far
- totalExitedValidators: uint64
- incremental counter of all exited validators for the NO so far

- totalAddedValidators: uint64
- incremental counter of all added to the NO validators so far
- totalDepositedValidators: uint64
- incremental counter of all deposited validators for the NO so far
- targetValidatorsCount: uint256
- target value for the number of validators for the NO. If the current active number of validators is below the value, the excess ones will be requested to exit. Allocation of deposits above the target value is prohibited. The exiting works only ifisTargetLimitActive
- is true. The0
- value will cause exit requests issued for all deposited validators of the NO.
- isTargetLimitActive: bool
- flag whether NO validators number is target-limited (seetargetValidatorsCount
-)
- stuckValidatorsCount: uint256
- number of stuck keys delivered by the oracle report
- refundedValidatorsCount: uint256
- number of refunded validators
- depositableValidatorsCount: uint256
- number of depositable validators

The values can be viewed by means ofgetNodeOperator() andgetNodeOperatorSummary() .

Except for the function listed below, the contract has methods accessible only byStakingRouter (holder ofSTAKING_ROUTER_ROLE). These functions are called internally in the course ofAccountingOracle report.

View Methods

getRewardsDistribution()

Returns the rewards distribution proportional to the effective stake for each node operator

function

getRewardsDistribution (uint256 _totalRewardShares)

returns

(address [ ] recipients , uint256 [ ] shares , bool [ ] penalized) Name Type Description _totalRewardShares uint256 Total amount of reward shares to distribute

getActiveNodeOperatorsCount()

Returns the number of active node operators.

function

getActiveNodeOperatorsCount ( )

returns

(uint256)

getNodeOperator()

Returns the node operator by id.

function

getNodeOperator (uint256 _nodeOperatorId ,

bool _fullInfo)

returns

(bool active , string name , address rewardAddress , uint64 totalVettedValidators , uint64 totalExitedValidators , uint64 totalAddedValidators , uint64 totalDepositedValidators) Name Type Description _nodeOperatorId uint256 Node operator id _fullInfo bool If true, name will be returned as well

getTotalSigningKeyCount()

Returns the total number of signing keys of the node operator.

function

getTotalSigningKeyCount (uint256 _nodeOperatorId)

returns

(uint256) Name Type Description _nodeOperatorId uint256 Node operator id

getUnusedSigningKeyCount()

Returns the number of usable signing keys of the node operator.

function

getUnusedSigningKeyCount (uint256 _nodeOperatorId)

returns

(uint256) Name Type Description _nodeOperatorId uint256 Node operator id

getSigningKey()

Returns n-th signing key of the node operator.

function

getSigningKey (uint256 _nodeOperatorId ,

uint256 _index)

returns

(bytes key , bytes depositSignature , bool used) Name Type Description _nodeOperatorId uint256 Node operator id _index uint256 Index of the key, starting with 0 Returns:

Name Type Description key bytes Key depositSignature bytes Signature needed for adepositContract.deposit call used bool Flag indication if the key was used in the staking

getSigningKeys()

Returns subset of the signing keys of the node operator corresponding to the specified range[_offset, _offset + _limit) . If the range is out of bound of range[0,) reverts withOUT_OF_RANGE error.

function

getSigningKeys (uint256 _nodeOperatorId ,

uint256 _offset ,

uint256 _limit)

returns

(bytes

memory pubkeys , bytes

memory signatures , bool [ ]

memory used) Name Type Description _nodeOperatorId uint256 Node operator id _offset uint256 Offset of the key in the array of all NO keys (0 means the first key,1 the second, etc _limit uint256 Number of keys to return Returns:

Name Type Description pubkeys bytes Keys concatenated into the bytes batch:[48 bytes key | 48 bytes key | ...] signatures bytes Signatures needed for adeposit_contract.deposit call, concatenated as[96 bytes | 96 bytes | ...] used bool[] Array of flags indicated if the key was used in the staking

getNodeOperatorsCount()

Returns the total number of node operators.

function

getNodeOperatorsCount ( )

returns

(uint256)

getNonce()

Returns a counter that increments whenever the deposit data set changes. Namely, it increments every time when for a node operator:

- staking limit changed;
- target validators limit changed;
- stuck validators count changed;
- exited validators count changed;
- validator signing keys added/removed;
- penalty is cleared;
- ready to deposit keys invalidated (due to withdrawal credentials change or due to manual invalidation by call ofinvalidateReadyToDepositKeysRange
-);
- ether deposited.

function

getNonce ( )

view

returns

(uint256)

getType()

Returns the type of the staking module.

function

getType ( )

view

returns

(bytes32)

getStakingModuleSummary()

Returns some statistics of the staking module.

function

getStakingModuleSummary ( )

view

returns

(uint256 totalExitedValidators , uint256 totalDepositedValidators , uint256 depositableValidatorsCount) Name Type Description totalExitedValidators uint256 Total number of exited validators totalDepositedValidators uint256 Total number of deposited validators depositableValidatorsCount uint256 Number of validators which can be deposited

getNodeOperatorIsActive()

Returns if the node operator with given id is active.

function

getNodeOperatorIsActive (uint256 _nodeOperatorId)

view

returns

(bool) Name Type Description _nodeOperatorId uint256 Node operator id

getNodeOperatorIds()

Returns up to_limit node operator ids starting from the_offset .

function

getNodeOperatorIds (uint256 _offset ,

uint256 _limit)

view returns

(uint256 [ ]

memory nodeOperatorIds) Name Type Description _offset uint256 Offset of the first element of the range _limit uint256 Max number of NO ids to return

getNodeOperatorSummary()

Returns some statistics of the node operator.

function

getNodeOperatorSummary (uint256 _nodeOperatorId)

view

returns

(bool isTargetLimitActive , uint256 targetValidatorsCount , uint256 stuckValidatorsCount , uint256 refundedValidatorsCount , uint256 stuckPenaltyEndTimestamp , uint256 totalExitedValidators , uint256 totalDepositedValidators , uint256 depositableValidatorsCount) Name Type Description isTargetLimitActive bool Is limiting target active validators count for NO enabled targetValidatorsCount uint256 Target validators count for full description seeparameters section stuckValidatorsCount uint256 Number of stuck keys from oracle report refundedValidatorsCount uint256 Number of refunded keys stuckPenaltyEndTimestamp uint256 Extra penalty time after stuck keys refunded totalExitedValidators uint256 Number of keys in the EXITED state of the NO for all time totalDepositedValidators uint256 Number of keys of the NO which were in DEPOSITED state for all time depositableValidatorsCount uint256 Number of validators which can be deposited

getStuckPenaltyDelay()

Returns value of the stuck penalty delay (in seconds). This parameter defines how long a penalized NO stays in penalty state after the stuck keys were refunded.

function

getStuckPenaltyDelay ( )

view

returns

(uint256)

isOperatorPenalized()

Returns flag whether the NO is penalized.

function

isOperatorPenalized (uint256 _nodeOperatorId)

view

returns

(bool)

isOperatorPenaltyCleared()

Returns flag whether the NO penalty is cleared.

function

isOperatorPenalized (uint256 _nodeOperatorId)

view

returns

(bool)

getLocator()

Returns the address ofLidoLocator .

function

getLocator ( )

view

returns

(ILidoLocator)

Methods

addNodeOperator()

Add node operator named_name with reward address_rewardAddress and staking limit = 0.

Executed on behalf of holder ofMANAGE_NODE_OPERATOR_ROLE role.

function

addNodeOperator (string _name , address _rewardAddress)

returns

(uint256 id) Name Type Description _name string Human-readable name _rewardAddress address Address which receives stETH rewards for this operator Returns:

Name Type Description id uint256 A unique key of the added operator

activateNodeOperator()

Activates deactivated node operator with given id.

Executed on behalf of holder ofMANAGE_NODE_OPERATOR_ROLE role.

note Increases the validators keys nonce function

activateNodeOperator (uint256 _nodeOperatorId) Name Type Description _nodeOperatorId uint256 Node operator id

deactivateNodeOperator()

Deactivates active node operator with given id.

Executed on behalf of holder ofMANAGE_NODE_OPERATOR_ROLE role

note Increases the validators keys nonce function

deactivateNodeOperator (uint256 _nodeOperatorId) Name Type Description _nodeOperatorId uint256 Node operator id

setNodeOperatorName()

Change human-readable name of the node operator with given id.

Executed on behalf of holder ofMANAGE_NODE_OPERATOR_ROLE role.

function

setNodeOperatorName (uint256 _nodeOperatorId ,

string _name) Name Type Description _nodeOperatorId uint256 Node operator id _name string Human-readable name

setNodeOperatorRewardAddress()

Change reward address of the node operator with given id.

Executed on behalf of holder ofMANAGE_NODE_OPERATOR_ROLE role.

function

setNodeOperatorRewardAddress (uint256 _nodeOperatorId ,

address _rewardAddress) Name Type Description _nodeOperatorId uint256 Node operator id _rewardAddress address New reward address

setNodeOperatorStakingLimit()

Set the maximum number of validators to stake for the node operator with given id.

Executed on behalf of holder ofSET_NODE_OPERATOR_LIMIT_ROLE role.

note Current implementation preserves invariant:depositedSigningKeysCount <= vettedSigningKeysCount <= totalSigningKeysCount . If_vettedSigningKeysCount out of range[depositedSigningKeysCount, totalSigningKeysCount] , the new vettedSigningKeysCount value will be set to the nearest range border. note Increases the validators keys nonce function

setNodeOperatorStakingLimit (uint256 _nodeOperatorId ,

uint64 _vettedSigningKeysCount) Name Type Description _nodeOperatorId uint256 Node operator id to set staking limit for _vettedSigningKeysCount address New staking limit of the node operator

addSigningKeys()

Add_keysCount validator signing keys to the keys of the node operator_nodeOperatorId .

Can be executed for the given NO if called from the NO's reward address or by the holder ofMANAGE_SIGNING_KEYS role.

note Along with each keypubkey a signatures must be provided for the(pubkey, withdrawal_credentials, 32000000000) message. For the details see thekeys section in NO guide .

Given that information, the contract'll be able to call deposit_contract.deposit on-chain. note Increases the validators keys nonce function

addSigningKeys (uint256 _nodeOperatorId , uint256 _keysCount , bytes _publicKeys , bytes _signatures) Name Type Description _nodeOperatorId uint256 Node operator id _keysCount uint256 Number of signing keys provided _publicKeys bytes Several concatenated validator signing public keys _signatures bytes Several concatenated signatures for the DepositContract messages see thekeys section in NO guide

removeSigningKeys()

Removes an_keysCount of validator signing keys starting from_fromIndex of operator_nodeOperatorId usable keys.

Can be executed for the given NO if called from the NO's reward address or by the holder ofMANAGE_SIGNING_KEYS role.

Keys removing from the last index to the highest one, so we won't get outside the array.

note Increases the validators keys nonce function

removeSigningKeys (uint256 _nodeOperatorId ,

uint256 _fromIndex ,

uint256 _keysCount) Name Type Description _nodeOperatorId uint256 Node operator id _fromIndex uint256 Index of the key, starting with 0 _keysCount uint256 Number of keys to remove

invalidateReadyToDepositKeysRange()

Invalidates all unused validators keys for node operators in the given range. Executed on behalf of holder ofMANAGE_NODE_OPERATOR_ROLE role.

function

invalidateReadyToDepositKeysRange (uint256 _indexFrom ,

uint256 _indexTo) Name Type Description _indexFrom uint256 the first index (inclusive) of the NO to invalidate keys for _indexTo uint256 The last index (inclusive) of the NO to invalidate keys for

clearNodeOperatorPenalty()

Clears penalty state for the NO if it is suitable for clearing. The penalty state is switched automatically upon oracle report if the conditions are met (e.g., if penalty delay expired), but this function allows it to happen quicker. Can be called by anyone.

function

clearNodeOperatorPenalty (uint256 _nodeOperatorId)

external

returns

(bool) Name Type Description _nodeOperatorId uint256 Id of the NO

setStuckPenaltyDelay()

Sets the stuck penalty delay parameter.

Add node operator named_name with reward address_rewardAddress and staking limit = 0. Executed on behalf of holder ofMANAGE_NODE_OPERATOR_ROLE role.

function

setStuckPenaltyDelay (uint256 _delay) Name Type Description _delay uint256 Stuck penalty delay in secondsEdit this page Previous StakingRouter Next WithdrawalQueueERC721