Burner

- Source Code
- Deployed Contract

The contract provides a way for Lido protocol to burn stETH token shares as a means to finalize withdrawals, penalize untimely exiting node operators, and, possibly, cover losses in staking.

It relies on therebasing nature of stETH. TheLido contract calculates user balance using the following equation:balanceOf(account) = shares[account] * totalPooledEther / totalShares . Therefore, burning shares (e.g. decreasing thetotalShares amount) increases stETH holders' balances.

It's presumed that actual shares burning happens inside theLido contract as a part of theAccountingOracle report.Burner provides a safe and deterministic way to incur a positive stETH token rebase by gradually decreasingtotalShares that can be correctly handled by 3rd party protocols integrated with stETH.

Burner accepts burning requests in the following two ways:

- Lockingsomeone's pre-approved
- stETH by the caller with the assignedREQUEST_BURN_SHARES_ROLE
- ;
- Lockingcaller-provided
- stETH with theREQUEST_BURN_MY_STETH_ROLE
- assigned role.

Those burn requests are initially set by the contract to a pending state. Actual burning happens as part of an oracle (AccountingOracle) report handling byLido to prevent additional fluctuations of the existing stETH token rebase period (~24h).

We also distinguish two types of shares burn requests:

- request tocover
- a slashing event (e.g. decreasing of the total pooled ETH amount
- between the two consecutive oracle reports);
- request to burn shares for any other cases (non-cover
-).

The contract has two separate counters for the burnt shares: cover and non-cover ones. The contract is exclusively responsible for the stETH shares burning byLido and burning allowed only from the contract's own balance only.

Shares burnt counters

The contract keeps count of all shares ever burned by way of maintaining two internal counters:totalCoverSharesBurnt andtotalNonCoverSharesBurnt for cover and non-cover burns, respectively. These counters are increased when actual stETH burn is performed as part of the Lido Oracle report.

This makes it possible to split any stETH rebase into two sub-components: the rewards-induced rebase and cover application-induced rebase, which can be done as follows:

1. Before the rebase, store the previous values of both counters, as well as the value of stETH share price:
2. prevCoverSharesBurnt
3. =
4. Burner
5. .
6. totalCoverSharesBurnt
7. (
8.)
9. prevSharePrice
10. =
11. stETH
12. .
13. totalSupply
14. (
15.)
16. /
17. stETH
18. .
19. getTotalShares
20. (

)
After the rebase, perform the following calculations:
sharesBurntFromOldToNew
=
Burner
.
totalCoverSharesBurnt
(
)
-
prevCoverSharesBurnt
;
newSharePriceAfterCov
=
stETH
.
totalSupply
(
)
/
(
stETH
.
getTotalShares
(
)
+
sharesBurntFromOldToNew
)
;
newSharePrice
=
stETH
.
totalSupply
(
)
/
stETH
.
getTotalShares
(
)
;
// rewards-induced share price increase
rewardPerShare
=
newSharePriceAfterCov
-
prevSharePrice
;
// cover-induced share price increase
nonRewardSharePriceIncrease
=
newSharePrice
-
prevSharePrice
-
rewardPerShare
;

View methods

getCoverSharesBurnt()

Returns the total cover shares ever burnt.

function

getCoverSharesBurnt ( )

external

view

returns

(uint256)

getNonCoverSharesBurnt()

Returns the total non-cover shares ever burnt.

function

getNonCoverSharesBurnt ( )

external

view

returns

(uint256)

getExcessStETH()

Returns the stETH amount belonging to the burner contract address but not marked for burning.

function

getExcessStETH ( )

external

view

returns

(uint256)

getSharesRequestedToBurn()

Returns numbers of cover and non-cover shares requested to burn.

function

getSharesRequestedToBurn ( )

external

view

returns

(uint256 coverShares ,

uint256 nonCoverShares)

Methods

requestBurnMyStETHForCover()

Transfers stETH tokens from the message sender and irreversibly locks these on the burner contract address. Internally converts tokens amount into underlying shares amount and marks the converted shares amount for cover-backed burning by increasing the internalcoverSharesBurnRequested counter.

function

requestBurnMyStETHForCover (uint256 _stETHAmountToBurn)

external note Reverts if any of the following is true:

- msg.sender
- is not a holder of theREQUEST_BURN_MY_STETH_ROLE
- role;
- no stETH provided (_stETHAmountToBurn == 0
-);
- no stETH transferred (allowance exceeded).

Parameters

Name Type Description _stETHAmountToBurn uint256 stETH tokens amount (not shares amount) to burn

requestBurnSharesForCover()

Transfers stETH shares from_from and irreversibly locks these on the burner contract address. Internally marks the shares amount for cover-backed burning by increasing the internalcoverSharesBurnRequested counter.

Can be called only by a holder ofREQUEST_BURN_SHARES_ROLE . After Lido V2 upgrade not actually called by any contract and supposed to be called by Lido DAO Agent in case of a need for cover.

function

requestBurnSharesForCover (address _from ,

uint256 _sharesAmountToBurn) note Reverts if any of the following is true:

- msg.sender
- is not a holder of theREQUEST_BURN_SHARES_ROLE
- role;
- no stETH shares provided (_sharesAmountToBurn == 0
-);
- no stETH shares transferred (allowance exceeded).

Parameters

Name Type Description _from address address to transfer shares from _sharesAmountToBurn uint256 shares amount (not stETH tokens amount) to burn

requestBurnMyStETH()

Transfers stETH tokens from the message sender and irreversibly locks these on the burner contract address. Internally converts tokens amount into underlying shares amount and marks the converted amount for non-cover backed burning by increasing the internalnonCoverSharesBurnRequested counter.

function

requestBurnMyStETH (uint256 _stETHAmountToBurn)

external note Reverts if any of the following is true:

- msg.sender
- is not a holder of theREQUEST_BURN_MY_STETH_ROLE
- role;
- no stETH provided (_stETHAmountToBurn == 0
-);
- no stETH transferred (allowance exceeded).

Parameters

Name Type Description _stETHAmountToBurn uint256 stETH tokens amount (not shares amount) to burn

requestBurnShares()

Transfers stETH shares from_from and irreversibly locks these on the burner contract address. Internally marks the shares amount for non-cover backed burning by increasing the internalnonCoverSharesBurnRequested counter.

Can be called only by a holder of theREQUEST_BURN_SHARES_ROLE role which after Lido V2 upgrade is either Lido orNodeOperatorsRegistry .Lido needs this to request shares locked on theWithdrawalQueueERC721

andNodeOperatorsRegistry needs it to request burning shares to penalize the rewards of misbehaving node operators.

function

requestBurnShares (address _from ,

uint256 _sharesAmountToBurn) note Reverts if any of the following is true:

- msg.sender
- is not a holder ofREQUEST_BURN_SHARES_ROLE
- role;
- no stETH shares provided (_sharesAmountToBurn == 0
-);
- no stETH shares transferred (allowance exceeded).

Parameters

Name Type Description _from address address to transfer shares from _sharesAmountToBurn uint256 shares amount (not stETH tokens amount) to burn

recoverExcessStETH()

Transfers the excess stETH amount (e.g. belonging to the burner contract address but not marked for burning) to the Lido treasury address (theDAO Agent contract) set upon the contract construction.

Does nothing if thegetExcessStETH view func returns 0 (zero), i.e. there is no excess stETH on the contract's balance.

function

recoverExcessStETH ( )

external

recoverERC20()

Transfers a given amount of an ERC20-token (defined by the provided contract address) belonging to the burner contract address to the Lido treasury (theDAO Agent contract) address.

function

recoverERC20 (address _token ,

uint256 _amount)

external note Reverts if any of the following is true:

- _amount
- value is 0 (zero);
- _token
- address is 0 (zero);
- _token
- address equals to thestETH
- address (userecoverExcessStETH
- instead).

Parameters

Name Type Description _token address ERC20-compatible token address to recover _amount uint256 Amount to recover

recoverERC721()

Transfers a given ERC721-compatible NFT (defined by the contract address) belonging to the burner contract address to the Lido treasury (theDAO Agent) address.

function

recoverERC721 (address _token ,

uint256 _tokenId)

external note Reverts if any of the following is true:

- _token
- address is 0 (zero);
- _token
- address equals to thestETH
- address (userecoverExcessStETH
- instead).

Parameters

Name Type Description _token address ERC721-compatible token address to recover _tokenId uint256 Token id to recover

commitSharesToBurn()

Marks previously requested to burn cover and non-cover share as burnt. EmitsStETHBurnt event for the cover and non-cover shares marked as burnt.

This function is called by theLido contract together with (i.e., the same tx) performing the actual shares burning.

If_sharesToBurn is 0 does nothing.

function

commitSharesToBurn (uint256 _sharesToBurn)

external note Reverts if any of the following is true:

- msg.sender
- address is NOT equal to thestETH
- address;
- _sharesToBurn
- is greater than the cover plus non-cover shares requested to burn.

Parameters

Name Type Description _sharesToBurn uint256 Amount of cover plus non-cover shares to mark as burntEdit this page
Previous DepositSecurityModule Next LidoExecutionLayerRewardsVault