小额质押

**问题**

基于权益证明（PoS）协议的项目的质押是将一个人的代币长期锁定在一个项目中，并期望获得固定的、预定的质押奖励作为回报。虽然它像债券一样保证了质押代币的回报，但它也限制了从 DeFi 生态系统中对这些代币产生更高回报的机会。

**流动性质押**

流动性质押允许使用其他交易机会让用户两全其美 - 您的质押代币的奖励，以及新交易机会的回报。流动性质押通过以下方式引入了各种基本优势：

* 使质押过程变得简单 - 无需担心硬件设置和维护;
* 使用户想要的小额存款获得奖励成为可能
* 为其他应用程序和协议提供构建。流动性质押提供了最大化潜力的机会;
* 提供交易所质押、单独质押和其他半托管和去中心化协议的替代方案，甚至包括交易所质押。

### 奖励分配模式

可以在存款后的 24 小时内获得质押奖励，而无需等待验证器激活。

### 费用

对质押奖励收取 10% 的费用。

### POOL:

流动质押池，核心合约负责：

* 接受用户的质押，缓冲并铸造相应数量的流动代币
* 根据收到的预言机报告和协议的当前状态进行适当的核算
* 将提款、优先费和 MEV 从各自的保险库收集到缓冲区中
* 申请费用和分配奖励
* 将缓存的以太币进一步传递到[StakingRouter](https://docs.lido.fi/contracts/staking-router)或[WithdrawalQueueERC721](https://docs.lido.fi/contracts/withdrawal-queue-erc721)

其他合同具有以下责任：

* [WithdrawalQueueERC721](https://docs.lido.fi/contracts/withdrawal-queue-erc721): 先进先出的提款请求队列和相应的 NFT
* [StakingRouter](https://docs.lido.fi/contracts/staking-router): 管理[质押](https://docs.lido.fi/contracts/staking-router)模块和在他们之间分配股份
* [OracleReportSanityChecker](https://docs.lido.fi/contracts/oracle-report-sanity-checker): 帮助程序验证预言机报告的参数并平滑令牌变基
* [Burner](https://docs.lido.fi/contracts/burner): 一个vault包含预言机报告的应该燃烧的stETH
* [WithdrawalVault](https://docs.lido.fi/contracts/withdrawal-vault): 一个vault收集部分和全部来自信标链的提款
* [LidoExecutionLayerRewardsVault](https://docs.lido.fi/contracts/lido-execution-layer-rewards-vault): 一个vault收取来自池验证者的优先费和 MEV 奖励
* [AccountingOracle](https://docs.lido.fi/contracts/accounting-oracle): 预言机集合，给协议的不断出具报告

## Submit[​](https://docs.lido.fi/contracts/lido/#submit)

要参加质押，a 用户可以向合约发送一些 ETH，相同数量的steth代币将被铸造到发送地址。提交的以太币累积在缓冲区，可以进一步传递到 [WithdrawalQueueERC721](https://docs.lido.fi/contracts/withdrawal-queue-erc721) 以满足提款请求或向[StakingRouter](https://docs.lido.fi/contracts/staking-router) 存入作为新的 validator stake。

要提取底层ETH，用户可以使用[WithdrawalQueueERC721](https://docs.lido.fi/contracts/withdrawal-queue-erc721)合约或在二级市场上交换代币（这可能是更便宜，更快捷的选择）。

## Deposit[​](https://docs.lido.fi/contracts/lido/#deposit)

用户提交的以太存储在缓冲区中，以后可用于提款或进一步传递到 [StakingRouter](https://docs.lido.fi/contracts/staking-router)用作validator的存款。它异步发生并使用[DepositSecurityModule](https://docs.lido.fi/contracts/deposit-security-module) 作为防护来防止存款抢跑漏洞。

## Redeem[​](https://docs.lido.fi/contracts/lido/#redeem)

token可以通过协议使用 [WithdrawalQueueERC721](https://docs.lido.fi/contracts/withdrawal-queue-erc721)合约兑换为以太币。

## Rebase[​](https://docs.lido.fi/contracts/lido/#rebase)

当预言机报告产生时，令牌的供应增加或减少，基于在信标链上的质押奖励（或削减惩罚），执行层奖励或满足的提款请求。

重定基数机制是通过shares概念实现的。相对于存储带有帐户余额的mapping，我们存储归帐户所有的在总池中的份额。账户余额的计算方法如下：

balanceOf(account) = shares[account] \* totalPooledEther / totalShares

* shares- 用户帐户的shares mapping。每次用户存入以太币时，它都是 转换为shares并添加到当前用户shares金额中。
* totalShares- mapping中所有账户的shares总和
* totalPooledEther- 协议拥有的三种类型的以太币的总和：
  + buffered balance - 以太币存储在合约中，尚未存入或已锁定提款
  + transient balance - 以太币提交给官方存款合约，但未提交但在信标状态下可见
  + beacon balance - 验证人账户上的以太币总量。此值由预言机报告，对STETH总供应量的影响最大改变

## Oracle report[​](https://docs.lido.fi/contracts/lido/#oracle-report)

预言机报告，通常 （但不保证）每天一次为协议提供不易于在链上处理，但需要精确计算的数据。它包括一些信标链统计信息以及有效的关于报告块EL-side values和履行待定提款请求所需的决策数据。

* Beacon chain stats:
  + the total number of validators managed by the pool
  + the total balance of validators managed by the pool
* Historical EL values:
  + withdrawal vault balance
  + execution layer rewards vault balance
  + burner stETH shares balance
* Withdrawal-related data
  + requests in the queue to be finalized
  + share rate to be used for finalization

Oracle 报告通过 9 个简单步骤进行处理：

1. 记住协议余额增量更新所需的预状态
2. 使用 [OracleReportSanityChecker](https://docs.lido.fi/contracts/oracle-report-sanity-checker) 验证报告数据
3. 计算要在[WithdrawalQueueERC721](https://docs.lido.fi/contracts/withdrawal-queue-erc721)上锁定的以太币数量，并将要销毁的相应份额数量移动到[Burner](https://docs.lido.fi/contracts/burner)
4. 使用[OracleReportSanityChecker](https://docs.lido.fi/contracts/oracle-report-sanity-checker)计算可以从[LidoExecutionLayerRewardsVault](https://docs.lido.fi/contracts/lido-execution-layer-rewards-vault)和[WithdrawVault](https://docs.lido.fi/contracts/withdrawal-vault)中提取的以太币数量以及可以从[Burner](https://docs.lido.fi/contracts/burner)中烧毁shares数量，以避免变基抢跑。
5. 从金库中收集计算出的以太币数量并处理最终确定提款请求：将请求的以太币发送到[提款队列](https://docs.lido.fi/contracts/withdrawal-queue-erc721)
6. 为提款或保险要求，从 [Burner](https://docs.lido.fi/contracts/burner) 销毁先前请求的shares
7. 为各方分配奖励和铸造新的stETH的协议费
8. 通过通知观察者来完成令牌变基（发出事件并调用外部接收器（如果有）
9. 报告后,对报告提供的份额率进行健全性检查

因此，协议报告的可观察结果如下：

* 队列中的提款请求已满足
* 以太币从提款和EL奖励金库中收集到缓冲区
* CL余额根据报告更新
* 奖励在质押者、质押模块和协议库之间分配

## Standards[​](https://docs.lido.fi/contracts/lido/#standards)

Contract implements the following Ethereum standards:

* [ERC-20: Token Standard](https://eips.ethereum.org/EIPS/eip-20)
* [ERC-2612: Permit Extension for ERC-20 Signed Approvals](https://eips.ethereum.org/EIPS/eip-2612)
* [EIP-712: Typed structured data hashing and signing](https://eips.ethereum.org/EIPS/eip-712)
* [ERC-1271: Standard Signature Validation Method for Contracts](https://eips.ethereum.org/EIPS/eip-1271)

## stETH vs wstETH

在用户存款后，发行stToken，它代表存入的代币以及通过存款质押产生的所有奖励和惩罚。与质押资金不同，该 stToken 具有流动性——它可以在各方之间自由转移。为用户提供的 stETH 等于质押的金额。

stTokens有两个版本，即stETH和wstETH。 两者都是可替代的代币，但它们以不同的方式反映了累积的质押奖励。stETH实现了变基机制，这意味着stETH余额会定期增加。相反，wstETH 余额是恒定的，而代币的价值最终会增加（以 stETH 计价）。在任何时候，任何数量的stETH都可以通过包装转换为wstETH，反之亦然，因此代币有效地共享流动性。

## stETH

stETH 是一种可变基的 ERC-20 代币，代表与质押的以太币。与质押以太币不同，它是流动的，可以转移、交易或在 DeFi 应用程序中使用。stETH 的总供应量反映了存入协议的以太币数量以及质押奖励，减去潜在的验证者惩罚。stETH 代币在以太币存款时以 1：1 的比例铸造。通常，当预言机报告信标链以太币余额更新时，stETH 代币余额每天都会重新计算。在变基时，stETH 余额更新会自动发生在持有 stETH 的所有地址上。

## stETH internals: share mechanics

每日变基会导致 stETH 代币余额发生变化。此机制通过shares实现。 这是一个基本单位，代表stETH持有者在协议控制的以太币总量中的份额。当发生新的存款时，新shares被铸造以反映协议控制的以太币的份额已添加到池中。

计算公式：

shares[account] = balanceOf(account) \* totalShares / totalPooledEther

## wstETH

wstETH是一种 ERC20 代币，代表账户在 stETH 总供应量中的份额shares。对于 wstETH 而言，1 wei 的share等于 1 wei 的余额。wstETH 余额只能在转账、铸造和燃烧时更改。wstETH余额不会变基，wstETH以stETH计价的价格会发生变化。 在任何给定时间，任何持有 wstETH 的人都可以以固定汇率将其转换为 stETH，反之亦然。在任何给定时刻，每个人的费率都是一样的。通常，当stETH进行变基时，费率每天更新一次。

## 提款

要从协议中获取以太币，需要继续执行以下步骤：

* 请求提款，将您的 steth 锁定在队列中并接收代表您在队列中的位置的 NFT
* 等待，直到请求由 Oracle 报告完成并可认领
* 领取你的以太币，烧掉 NFT

请求大小应至少为 100 wei（以 stETH 为单位），最多为 **1000 stETH**。较大的金额应在多个请求中提取，这些请求可以通过协议内 API 进行批处理。一旦提出要求，提款不能取消。提款 NFT 可以转移到不同的地址，一旦最终确定，新所有者将能够领取请求的提款。

处理提款的最终用户合同是，它执行ERC721标准。NFT 代表提款队列中的头寸，可以在请求完成后领取。

**检查提款状态**

* 可以通过调用返回 NFT ID 数组来检查所有者的所有提款请求
* 要检查特定 NFT 的状态，您可以调用它返回一个 [WithdrawalRequestStatus](https://github.com/lidofinance/lido-dao/blob/feature/shapella-upgrade/contracts/0.8.9/WithdrawalQueueBase.sol#L67-L81) 结构数组。

**声称**

要领取以太币，需要：

* claimWithdrawal(uint256 \_requestId)代表 NFT 所有者使用 NFT ID
* claimWithdrawals(uint256[] \_requestIDs, uint256[] \_hints)如果您想批量申请多次提款

# AccountingOracle

AccountingOracle是一种合约，预言机在共识层发送由我们控制的地址余额。 余额可能会因为奖励积累而上升，也可能由于削减和质押惩罚而下降。

AccountingOracle的其他主要职责：更新退出和pending的验证器，完成提款请求。

Oracle 进程每frame（目前为 225 个区块，等于一天）推送其报告，并且当 报告的数量达到[“ quorum”](https://docs.lido.fi/contracts/accounting-oracle#getquorum)值，报告被推送到合约

# OracleReportSanityChecker

一些重要数据是在链下收集的，并通过Oracle合约在链上交付：[AccountingOracle，](https://docs.lido.fi/contracts/accounting-oracle)[ValidatorsExitBusOracle](https://docs.lido.fi/contracts/validators-exit-bus-oracle)。 由于预言机提供的数据对协议状态的影响很大，因此每个预言机的报告通过一组链上 [sanity checks](https://en.wikipedia.org/wiki/Sanity_check)。 为了简化负责处理 Oracle 报告的合同，所有健全性检查都收集在独立合同。

# StakingRouter

StakingRouter是一个质押模块的注册表，每个模块都封装了某个验证器子集。该合约将stake 分配给模块，分配协议费用并跟踪相关信息。

## StakingRouter [​](https://docs.lido.fi/contracts/staking-router#what-is-stakingrouter)

StakingRouter是用于质押模块的 top-level controller contract。每个质押模块都是一个合约，同样管理自己的验证者子集。

StakingRouter执行许多功能，包括：

* 维护质押模块的注册表，
* 将stake分配给模块，
* 分配协议费用。

## Module Management

### Registering a module

质押模块使用该函数注册，提供详细信息

* 模块名称：人类可读的名称;
* 已部署质押模块合约的地址;
* 目标份额;
* 模块费用，授予模块的质押奖励的百分比，
* 国库费，质押奖励的一定百分比将直接用于协议国库。

### Pausing modules

每个质押模块都有一个状态：一个确定模块是否可以执行存款并获得奖励的状态：

* Active，可以存款并获得奖励，
* DepositsPaused，不允许存款，但会收到奖励，
* Stopped，不能存款，也不会收到奖励。

### Exited and stuck validators[​](https://docs.lido.fi/contracts/staking-router#exited-and-stuck-validators)

当提款请求需求超过处于buffered 以太币以及预计奖励时，协议会发出信号，开始退出验证器以覆盖提款。在这方面，StakingRouter 区分了两种类型的验证器状态：

* [退出的](https://hackmd.io/zHYFZr4eRGm3Ju9_vkcSgQ?view)验证器，
* 卡住的验证器，即那些未能遵守退出信号的验证器。

StakingRouter 会跟踪这两种类型的验证者，以便正确分配权益并惩罚卡住的验证者。这些统计数据通过向合约提交数据的预言机保持最新。

## Stake allocation[​](https://docs.lido.fi/contracts/staking-router#stake-allocation)

StakingRouter 执行一项重要任务，即将可存入的以太币分配给质押模块。这种设计确保了新集成到系统中的模块的调节和控制增长。

### 存款

存款工作流程涉及在一笔交易中向 [DepositContract](https://ethereum.org/en/staking/deposit-contract/" \t "_blank) 提交一批 32 个以太币存款以及相关的验证人密钥。鉴于每个质押模块处理自己的存款，因此每个批量存款仅限于来自单个模块的密钥。

存款操作的核心是由链下软件引发的一系列合约调用。此软件收集guardian消息，以确认注册表中没有预先存在的可以[frontrunning vulnerability](https://github.com/lidofinance/lido-improvement-proposals/blob/develop/LIPS/lip-5.md)密钥。一旦达到必要的法定人数，软件就会将这些消息与模块标识符一起转发到DepositSecurityModule。该合约首先验证消息，然后启动存款功能，传递当前区块大小可以容纳的最大存款数量。随后，根据现有的存款缓冲区计算批处理中可以包含的最大存款数量，并触发 StakingRouter 的存款功能。然后，StakingRouter 确定缓冲以太币到模块的分配，该模块将在存款中使用其密钥，并最终执行批量存款。

该功能首先验证发送者的身份并检查提款凭据和质押模块的状态。在这些检查之后，它通过将当前时间戳和区块号记录为质押模块的最后存款时间和区块来更新合约的本地状态。然后，它会发出一个事件来记录存款交易，并检查存款价值是否与所需的总存款大小匹配。如果有存款，它会从质押模块合约中获取存款数据（公钥和签名）。然后，它使用获得的数据进行存款。最后，它通过比较存款交易前后合约的以太币余额，确认所有存入的ETH都已正确转移到合约中。

### Allocation algorithm

用于分配过程的算法旨在考虑各种因素，包括每笔交易的存款限额、准备存入的以太币数量、每个模块中的活动和可用密钥以及每个模块的目标份额。然后，它根据这些参数估计最大存款。该算法还识别所有模块的密钥限制，并从具有最少活动密钥的模块开始启动密钥分配。这种情况一直持续到没有更多的以太币要分配或所有模块都达到其容量为止。在该过程结束时，将返回任何剩余的以太币。

分配函数使用[MinFirstAllocationStrategy](https://github.com/lidofinance/lido-dao/blob/master/contracts/common/lib/MinFirstAllocationStrategy.sol)算法在不同的质押模块之间分配新的存款。

以下是该过程的细分：

1. 该函数作为输入，表示需要在质押模块之间分配的新存款量。
2. 它首先计算系统中的活动验证者总数并将质押模块的当前状态加载到缓存中。
3. 然后，它创建一个与质押模块数量相同大小的数组，该数组中的每个索引代表质押模块的当前分配（即该模块中当前活跃验证器的数量）。
4. 如果有可用的质押模块，则该函数进入分配过程：

a.它计算新的估计活跃验证者总数，将新存款添加到活跃验证者总数

b.它创建一个与质押模块数量相同大小的数组。该数组中的每个条目代表特定质押模块的最大容量，即该模块可以拥有的最大验证器数量。计算方法是：

* + 模块的目标验证器数，基于所需的目标份额stakingModulesCache[i].targetShare \* totalActiveValidators / TOTAL\_BASIS\_POINTS
  + 模块中当前活动验证程序和可用验证程序的总和。stakingModulesCache[i].activeValidatorsCount + stakingModulesCache[i].availableValidatorsCount

c. 最后，它调用函数allocate，传入MinFirstAllocationStrategy 、 allocations和 capacities。成功分配的金额存储在\_depositsToAllocate中。

总而言之，该函数使用算法在不同的质押模块之间分配新的存款（验证者），优先考虑填充数量最少的模块，同时考虑每个模块的目标份额和容量。然后返回结果分配和总分配金额以供进一步使用。

## Fee distribution

费用结构在每个模块中独立设置。费用结构有两个组成部分：模块费和国库费，两者都以百分比（基点）的形式指定。此外，利用 100 \* 10^18 的精度来支付费用，以防止算术运算截断小模块的费用。

由于协议当前不考虑每个验证者的性能，因此协议费用在模块之间按比例分配给活动验证者和指定的模块费用。还有一个边缘情况，即模块在其验证器仍处于活动状态时紧急停止。在这种情况下，模块费用将转移到国库，一旦模块重新上线，奖励将从国库返还给模块。

分布函数本身的工作原理如下：

1. 该函数首先将质押模块的当前状态加载到缓存中，并计算这些模块的数量。
2. 如果系统中没有质押模块或活动验证器，则返回空响应。
3. 否则，它将初始化数组以存储模块 ID 、奖励接收者的地址和每个接收者的费用。它还将设置一个常量，该常量表示构成 100% 费用的基本精度数。
4. 然后它循环遍历每个质押模块。对于至少有一个active validator的每个模块：
   * 将模块 ID 和收件人地址存储在相应的数组中。
   * 计算stakingModuleValidatorsShare ，这是属于此质押模块的活跃验证者总数的比例。
   * 计算质押模块的费用占可能的总费用的比例。如果模块未停止，此费用将存储在数组中。
   * 加上质押模块的费用和进入国库的费用，其中国库是中央资金池。
5. 遍历所有模块后，它会做出不超过 100% 的assertion （由totalFeeprecisionPoints表示）。
6. 如果存在没有active validators的质押模块，则会缩小stakingModuleIds 、recipients，stakingModuleFees 和数组以排除这些模块。

最后，该函数返回五个值：recipients、stakingModuleIds、stakingModuleFees、totalFee、precisionPoints。这些为调用者提供了如何在质押模块之间分配奖励的概述。

# WithdrawalQueueERC721

用于提款请求的 FIFO 队列和表示队列中位置的 NFT 实现。

使用[AccessControlEnumerable](https://github.com/lidofinance/lido-dao/blob/master/contracts/0.8.9/utils/access/AccessControlEnumerable.sol)合约的功能和一些[granular roles](https://docs.lido.fi/contracts/withdrawal-queue-erc721#roles)来限制对方法的访问。

## WithdrawalQueueERC721

它负责：

* 管理提款请求队列
* 将提款请求的最终状态作为 [AccountingOracle](https://docs.lido.fi/contracts/accounting-oracle)报告的一部分
* storing before and ether after the finalization
* 在索赔时将保留的以太币转让给用户

此外，该合约是 [ERC-721](https://eips.ethereum.org/EIPS/eip-721) NFT。此 NFT 应要求铸造并在索赔中烧毁。[ERC-4906](https://eips.ethereum.org/EIPS/eip-4906) 用于在请求的完成状态更改后立即更新元数据。

## Request

要请求提款，需要批准本合同的金额或签署[ERC-2612许可证](https://eips.ethereum.org/EIPS/eip-2612)，然后调用适当的方法。

请求的**最小**量为100 wei ，**最大值**为1000 eth 。更高的金额应被拆分为多个请求，这使我们能够避免因超大请求而堵塞队列。

在此调用期间，请求被放入队列中，并铸造相关的 NFT。用户在他们的代币留在队列中的一段时间内不会收到奖励。

## Finalization

提交提款请求后，只有在最终确定后才能索赔。[Accounting Oracle](https://docs.lido.fi/contracts/accounting-oracle) 报告最终确定一批提款请求，选择批次的前后时要考虑以下因素：

* 如果有足够的以太币来满足请求。eth可以从缓冲区获得。提款优先于存款，因此如果某些提款请求可以实现，进入缓冲区中的以太币不能存入信标链。
* 如果提款请求被放入队列（时间锁）已经过去了足够的时间
* 如果自提交提款请求以来信标链方面的协议有一些巨大的损失。

## Claim

当请求最终确定时，当前所有者可以要求，将保留数量的以太币转移到收款人的地址并销毁提款 NFT。

# WithdrawalVault

# WithdrawalVault

一个简单的合同，累积来自信标链的部分和全部提款。在accounting oracle报告期间，保管库清空

金库是可恢复的，因此任何 ERC-20 和 ERC-721 代币都可以转移到国库。

# Burner

该合约提供了一种销毁 stETH 代币份额的方法，作为完成提款的一种手段， 惩罚不合时宜的退出节点运营商，并可能弥补质押的损失。

它依赖于stETH的[rebasing](https://docs.lido.fi/contracts/lido" \l "rebasing) 性质。合约通过使用以下等式进行用户余额的计算： balanceOf(account) = shares[account] \* totalPooledEther / totalSharestotalShares

## Shares burnt counters

该合约通过维护两个内部计数器来记录所有烧毁的股票：分别用于cover and non-cover burns。 当实际的 stETH burn 作为Oracle 报告的一部分执行时，这些计数器会增加。

这使得可以将任何stETH变基分成两个子组件：the rewards-induced rebase and cover application-induced rebase，可以按如下方式完成：

1. 在变基之前，存储两个计数器的先前值，以及stETH股价的值：

prevCoverSharesBurnt = Burner.totalCoverSharesBurnt()  
prevSharePrice = stETH.totalSupply() / stETH.getTotalShares()

1. 变基后，执行以下计算：

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;

# LidoExecutionLayerRewardsVault

用于临时存储执行层 （EL） 奖励（MEV 和 tx 优先费）的保管库。

交易优先费和MEV奖励都是通过将合约地址指定为币库来收取的。此外，每当有效载荷构建者包含将 MEV 份额转移到有效载荷中的显式交易时，也会提取 MEV 奖励。因此，合约具有接受传入以太币的应付接收功能。

可以提取累积的奖励,在steth持有者之间分配，作为[Accounting Oracle](https://docs.lido.fi/contracts/accounting-oracle)报告的一部分。