TL;DR

Propose an architecture framework for the rebasable stETH token on L2 to be used across various Ethereum Layer 2 (L2) networks. The idea is to create a canonical version of stETH

on these networks, being wrapped on top of wstETH

(sort of ‘wrapped twice’). This approach will streamline the process of moving stETH

across Ethereum rollups, making use of the existing wstETH

infrastructure and liquidity venues on L2 networks.

As a practical example, explore a design of upgrading the wstETH

custom bridge on the Optimism network. The design relies on the previously presented [lido-l2

](https://github.com/lidofinance/lido-l2/tree/main/contracts/optimism) reference architecture and can be easily re-used for the OP-stack compatible networks or fairly simply adjusted for diverge rollup solutions.

Motivation on wstETH vs stETH

The wstETH token has been successfully integrated on various L2s: Optimism, Arbitrum, Base, zkSync Era, Linea, Mantle providing the ultimate easiness for DeFi integrations being a straightforward value-accruing token using the ‘lock-and-mint’ L1→L2 bridging approach.

In contrast to wstETH, stETH depends on the rate changes to recalculate each user’s balance (e.g., perform a rebase). There is an essential UX property: the stETH token balance represents the underlying Ether value expected to be redeemed 1:1 via the protocol upon a withdrawal request.

Beyond just updating an account’s balance in the wallet, having rebasable stETH on L2 hypothetically unlocks the following use cases and scenarios:

- staking/withdrawal requests originated from L2s with accuracy accounting for the stETH rebases that occurred during the bridging period
- performing gas payments in stETH

with amounts corresponding to ether on rollups that opted in to support token gas payments (e.g., following the Account Abstraction)

- cross-domain stETH

deposits/withdrawals (L1 and L2s) for CEXes

- custodians and service providers to support L2s directly (DeFi interactions, deposits/withdrawals) as they can charge fees with each rebase while users still see their balance changing each day
- once main routine user activity migrated from L1 to L2, there should be UX available as it was previously (rebasable token)

Abstract and flows

[

threeNewFlows

1913×1407 165 KB

](https://europe1.discourse-cdn.com/business20/uploads/lido/original/2X/e/e58452e98b584c6a44b8c2f3cfe97536f6eb71f9.png)

The ‘lock-and-mint’ wstETH bridge extension (i.e., [lido-l2

](https://github.com/lidofinance/lido-l2/tree/main/contracts/optimism)) introduces a set of the new bridging flows:

Bridging stETH from L1 to L2

User locks stETH, and the bridge contract wraps it into wstETH. The amount of wrapped tokens is then forwarded through the already implemented wstETH bridging path (e.g., wstETH on Optimism) with the intent to have stETH on the L2 end.

Therefore, the L2 bridge mints wstETH and wraps it back to stETH.

The design facilitates the L2 stETH bridged amount is being the same as was sent from L1 in most cases by attaching the token rate data with each bridging request.

Bridging back stETH from L2 to L1

The bridge transfers stETH from a user upon request, unwraps it to wstETH, and burns that wstETH on L2 side. The amount of burned tokens is then forwarded through the established path with the goal of receiving stETH on the L1 end. To achieve this, the L1 bridge, in turn, unlocks wstETH and unwraps it to stETH.

Since the withdrawal process can take up to a week (a common case for optimistic rollups), it's expected that the stETH amount will be different on the L1 side upon the withdrawal completion due to rebases happened during the withdrawal delay (i.e., challenge period) of a rollup, and user rewards/penalties will be attributed properly.

Forwarding the wstETH/stETH in-protocol rate from L1 to L2

Include the token rate as a part of the calldata in each L1 to L2 bridging request. This involves incorporating the L1 block.timestamp

and utilizing the information from wstETH | Lido Docs.

Furthermore, allow permissionless token rate updates to be sent from L1 to L2 with a zero token amount escrowing.

It's also suggested to push the token rate as a part of each AccountingOracle

report (upon the submitReportData

transaction execution).

Emergency flow

It's assumed that a rollup allows to perform force inclusion of the transaction in case of rogue or stale sequencer. To achieve this, the same approach of L1->L2

messaging is used, while the inclusion itselft is guaranteed when the sequencing window duration passes (e.g., 12h for Optimism, 24h for Arbitrum).

It's also assumed that general rollup failures and communication loss of L1->L2

messaging would mean that L2 blocks are not produced and the L2 state is not going forward. Also, the design presumes that no fraud L2 transactions are included.

Detailed specification

Please proceed to the LIP-22 text published on GitHub for further details.

What's next

- Stay tuned for announcements on the testnet deployments (ETA is ~end of March)
- The solution should be polished up, fortified, and underwent a rigorous security process before the mainnet proposal publicized (ETA is ~Q2 2024)
- The Network expansion workgroup will be looking for scaling opportunities for the rollups and integrations interested in stETH

adoption on L2 depending on the project traction and demand

Looking forward to hearing your feedback and considerations