DISCLAIMER. MEV extraction policy for Lido DAO is an ongoing topic and has not been finalized yet. This proposal is about additional tech spares.

Simple Summary

The on-chain allowed relays list is planned to be used by Node Operators participating in the Lido protocol after the Merge to extract MEV according to the expected Lido policies.

Motivation

It’s proposed that Node Operators use [MEV-Boost

](https://github.com/flashbots/mev-boost) infrastructure developed by Flashbots to support MEV extraction through the open market mechanics as a current PBS solution that has a market fit.

The proposed allowed list is intended to be a source of truth for the set of possible relays allowed to be used by Node Operators. In particular, Node Operators would use the contract to keep their software configuration up-to-date (setting the necessary relays once Lido DAO updates the set).

Mechanics

The contract represents simple registry storage.

Anyone can access the storage in a permissionless way through the disclosed view

methods or by collecting the emitted events covering all storage modifications.

Any modification of the set (i.e., internal storage modification) is allowed only bythe general Lido DAO governance process, which is implemented on-chain through the Aragon voting from the initial deployment stage. However, it’s still possible to assign a dedicated management entity for adding/removing relay items.

The proposed MEVBoostAllowedRelaysList

contract is non-upgradable, and its owner is intended to be initialized with theLido DAO Aragon Agent address on mainnet upon the deployment phase.

The dedicated management manager

entity is initialized with a zero address and can be set later by the contract’s owner to be able to add/remove relays. The possible candidate to bear such responsibilities without sacrificing governance security is Easy Track (in this case [EVMScriptExecutor (Easy Track)

](https://docs.lido.fi/deployed-contracts#easy-track) should be set as manager

).

Relay information structure

It’s proposed to use the following structure:

struct Relay: uri: String[MAX_STRING_LENGTH] operator: String[MAX_STRING_LENGTH] is_mandatory: bool description: String[MAX_STRING_LENGTH]

where:

- uri

is the relay’s URI to fetch the data

- operator

is the name of the entity running the relay

- is_mandatory

is supposed to distinguish between optional and mandatory relays

- description

is designated to store any additional info (unspecified at the current stage)

Adding relay

To add a new relay, owner

or manager

(if assigned) should call add_relay

method, passing all necessary relay structure's params described above.

Removing relay

To remove a relay, owner

or manager

(if assigned) should call remove_relay

method, passing the relay's uri

.

Updating relay

The proposed flow to update the relay's info is to re-add the relay (call remove_relay

and then add_relay

).

NB: the order of the relays in the list after re-adding might change (due to the array-based implementation).

Reading the current allowed relays

Anyone is allowed to call the following methods to retrieve the current relays set:

- get_relays_amount

to check how many relays are currently allowed by Lido DAO

- get_relays

to retrieve all currently allowed relays

- get_relay_by_uri

to retrieve the allowed relay details by uri

- get_allowed_list_version

to read lastly bumped version number of the allowed list

Allowed list versioning

The principal purpose of the proposed contract is to be used for generating configurations containing up-to-date allowed relays.

To facilitate config generation process, the contract contains the previously mentioned get_allowed_list_version

view method to check whether the lastly used allowed relays list was updated or not.

The version bumps on every add/remove operation and doesn't have additional semantical meaning and numbering schemes.

Events

All storage modification functions emit at least a single event containing all necessary data to reproduce the changes by external indexers:

- RelayAdded

(once a relay was allowed)

- RelayRemoved

(once a previously allowed relay was removed)

- RelaysUpdated

(once allowed list version bumped)

- OwnerChanged

(once the owner is changed)

- ManagerChanged

(once management entity is assigned or dismissed)

- ERC20Recovered

(once some ERC-20 tokens successfully recovered)

Permissions

The contract has a mutable owner set upon the deployment phase. It’s presumed that the owner will be set to the [Lido DAO Aragon Agent

](https://etherscan.io/address/0x3e40D73EB977Dc6a537aF587D48316feE66E9C8c) address. The owner can be changed by a call of change_owner

only by the current owner.

An additional manager

is allowed to add or remove relays (initially is set to zero address). Only the contract’s owner is allowed to assign or dismiss manager

.

Specification, security considerations, and reference implementation

Please proceed to the full LIP-17 text published on GitHub for further details.

Links

- Ongoing discussion draft for block proposer rewards
- Ethereum MEV Extraction and Rewards - Discussion & Policy Groundwork
- [Proposal] optimal MEV policy for Lido
- LIP-12: On-chain part of the rewards distribution after the Merge