MevBoostRelayAllowedList

- Source Code
- Deployed Contract (mainnet)
- Deployed Contract (goerli+prater)

MEV-Boost relay allowed list is a simple contract storing a list of relays that have been approved by DAO for use in MEV-Boost setups. The data from the contract is used to generate a configuration file that contains a list of relays that should be connected to by the Node Operators participating in Lido.

View methods

get_owner()

Retrieves the current contract owner.

@view @external def get_owner() -> address

get_manager()

Retrieves the current manager entity (returns zero address if no entity is assigned).

@view @external def get_manager() -> address

get_relays_amount()

Retrieves the current total amount of allowed relays.

@view @external def get_relays_amount() -> uint256

get_relays()

Retrieves all of the currently allowed relays.

@view @external def get_relays() -> DynArray[Relay, MAX_NUM_RELAYS]

get_relay_by_uri()

Retrieves the relay with the provided uri.

@view @external def get_relay_by_uri(relay_uri: String[MAX_STRING_LENGTH]) -> bool

Parameters:

Name Type Description relay_uri String[MAX_STRING_LENGTH] URI of the relay note Reverts if no relay found.

get_allowed_list_version()

Retrieves the current version of the allowed relays list.

@view @external def get_allowed_list_version() -> uint256:

Methods

add_relay()

Appends relay to the allowed list. Bumps the allowed list version.

@external def add_relay(uri: String[MAX_STRING_LENGTH], operator: String[MAX_STRING_LENGTH], is_mandatory: bool, description: String[MAX_STRING_LENGTH])

Parameters:

Name Type Description uri String[MAX_STRING_LENGTH] URI of the relay operator String[MAX_STRING_LENGTH] Name of the relay operator is_mandatory bool If the relay is mandatory for usage for Lido Node Operator description String[MAX_STRING_LENGTH] Description of the relay in free format note Reverts if any of the following is true:

- called by anyone except the owner or manager
- relay with provideduri
- already allowed
- uri
- is empty

remove_relay()

Removes the previously allowed relay from the set. Bumps the allowed list version.

@external def remove_relay(uri: String[MAX_STRING_LENGTH]):

Parameters:

Name Type Description uri String[MAX_STRING_LENGTH] URI of the relay note Reverts if any of the following is true:

- called by anyone except the owner or manager
- if relay with provideduri
- is not allowed
- uri
- is empty

change_owner()

Change current owner to the new one.

@external def change_owner(owner: address)

Parameters:

Name Type Description owner address Address of the new owner note Reverts if any of the following is true:

- called by anyone except the current owner
- owner
- is the current owner
- owner
- is zero address

set_manager()

Setsmanager as the current management entity.

@external def set_manager(manager: address)

Parameters:

Name Type Description manager address Address of the new manager note Reverts if any of the following is true:

- called by anyone except the current owner
- manager
- is equal to the previously set value
- manager
- is zero address

dismiss_manager()

Dismisses the current management entity.

@external def dismiss_manager() note Reverts if any of the following is true:

- called by anyone except the current owner
- nomanager
- was set previously

recover_erc20()

Transfers ERC20 tokens from the contract's balance to therecipient .

@external def recover_erc20(token: address, amount: uint256, recipient: address) note Reverts if any of the following is

true:

- called by anyone except the owner
- ERC20 transfer reverted
- recipient
- is zero address Edit this page Previous LidoExecutionLayerRewardsVault Next TRP VestingEscrow