2. ElevationOwnershipToken.sol

Purpose: Represents equity in Elevation Foundation assets. Distributes dividends from the treasury.

Solidity

```
// SPDX-License-Identifier: MIT
pragma solidity ^0.8.20;

import "@openzeppelin/contracts
/token/ERC20/extensions
/ERC20Burnable.sol";
import "@openzeppelin/contracts
/access/AccessControl.sol";

/**
 * @title ElevationOwnershipToken
(EOT)
 * @dev Represents equity stake in
Elevation Foundation. Tradable and
dividend-eligible.
 */
contract ElevationOwnershipToken is
ERC20Burnable, AccessControl {
    bytes32 public constant
TREASURY_ROLE =
keccak256("TREASURY_ROLE");

    mapping(address => uint256) public
lastClaimed;
    uint256 public dividendPerToken;
    uint256 public totalDistributed;

    event DividendsDistributed(uint256
amount);
    event DividendsClaimed(address
indexed user, uint256 amount);

    constructor() ERC20("Elevation
Ownership Token", "EOT") {

_grantRole(DEFAULT_ADMIN_ROLE,
msg.sender);
    }

    function mint(address to,
uint256 amount) external
onlyRole(DEFAULT_ADMIN_ROLE) {
        _mint(to, amount);
    }

    function distributeDividends()
external payable
onlyRole(TREASURY_ROLE) {
        require(totalSupply() > 0, "No
tokens in circulation");
        dividendPerToken += (msg.value *
1e18) / totalSupply();
        totalDistributed += msg.value;
        emit
DividendsDistributed(msg.value);
    }

    function claimDividends() external {
        uint256 owed =
getOwedDividends(msg.sender);
        require(owed > 0, "No dividends
owed");
        lastClaimed[msg.sender] =
dividendPerToken;

payable(msg.sender).transfer(owed);
        emit
DividendsClaimed(msg.sender, owed);
    }

    function getOwedDividends(address
user) public view returns (uint256) {
        uint256 delta = dividendPerToken -
lastClaimed[user];
        return (balanceOf(user) * delta) /
1e18;
    }

    receive() external payable {
        revert("Use distributeDividends");
    }
}
```