```
// SPDX-License-Identifier: MIT
pragma solidity ^0.8.0;

import "@openzeppelin/contracts/token/TRC20/ITRC20.sol";
import "@openzeppelin/contracts/access/Ownable.sol";

contract Cosmoflux is ITRC20, Ownable {
    uint256 private constant TOTAL_SUPPLY = 1_000_000_000 * (10 ** 18);

    constructor() ITRC20("Cosmoflux", "CSX") {
        _mint(msg.sender, TOTAL_SUPPLY); // All tokens minted at deployment
    }

    event Burn(address indexed burner, uint256 amount);
    event AutoBurn(address indexed system, uint256 amount);
    event GovernanceProposalCreated(address indexed proposer, string proposalDetails);

    // Automatic system-controlled burn (e.g., when content is created)
    function autoBurn(uint256 amount) external onlyOwner {
        require(balanceOf(address(this)) >= amount, "Insufficient CSX for auto-burn");
        _burn(address(this), amount);
        emit AutoBurn(address(this), amount);
    }

    // Future functionality: Scheduled burns upon reaching financial milestones
    function scheduledBurn(uint256 amount) external onlyOwner {
        require(balanceOf(address(this)) >= amount, "Insufficient CSX for scheduled burn");
        _burn(address(this), amount);
    }

    // Placeholder for future governance-controlled utility token introduction
    function proposeNewUtilityToken(string memory proposalDetails) external {
        emit GovernanceProposalCreated(msg.sender, proposalDetails);
    }
}
```