ERC20 token smart contract

Introduction

This document details a token smart contract that implements ERC20 interface, allowing token functionalities such as mint, burn, transfer. Besides, it has other supporting functions such as query for token symbol, supply, balance of an address.

This contract leverages Openzeppelin library for simplicity.

Smart contract functions

1. Constructor
    a. Deploy an ERC20 token with predefined name “MyToken” and symbol “MTK”
    b. Mint initial supply `100000000` to owner
2. Mint token
    a. mint(address to, uint256 value)
    b. Mint token to an address. Only owner can call this function.
3. Burn token
    a. burn(address from, uint256 value)
    b. Burn an amount of token from an address. Only owner can call this function. Holder must have sufficient token amount to burn.
4. Token transfer
    a. transfer(address to, uint256 value)
    b. Transfer a token amount from caller to receiver. Caller must have enough token balance to transfer.
5. Query
    a. symbol
    b. name
    c. totalSupply
    d. decimals
    e. owner

Deploy token

1. Go to smart contract root folder `smart-contracts`
2. Install dependencies `npm install`
3. Setup private key for deployment: `npx hardhat vars set SEPOLIA_PRIVATE_KEY`, paste private key used for deployment.
   Alternatively, set `SEPOLIA_PRIVATE_KEY` value at `hardhat.config.js`.
   Ensure the account has some testnet ETH for deployment.
4. Deploy token contract: `npx hardhat ignition deploy ./ignition/modules/MyToken.js --network sepolia`. Copy token address to be used for query at Token Service backend.

Other commands

1. Run test: `npx hardhat test`
2. Compile Solidity code into binary and interface: `npx hardhat compile`