require('dotenv').config();

const { DefenderRelayProvider, DefenderRelaySigner } = require('@openzeppelin/defender-relay-client');

const { ethers } = require('ethers');

async function main() {

const credentials = {

apiKey: process.env.DEFENDER\_API\_KEY,

apiSecret: process.env.DEFENDER\_API\_SECRET,

};

const provider = new DefenderRelayProvider(credentials);

const signer = new DefenderRelaySigner(credentials, provider, { speed: 'fast' });

// --- CONFIGURATION ---

// Replace with your deployed ELMC token address

const elmcAddress = process.env.ELMC\_CONTRACT\_ADDRESS; // Must be set in .env

// Amount of BNB to rescue (e.g., "0.1" for 0.1 BNB)

const amountToRescueBNB = ethers.parseEther(process.env.RESCUE\_BNB\_AMOUNT || "0.1"); // Default to 0.1 BNB if not set

// --- END CONFIGURATION ---

if (!elmcAddress) {

console.error("ERROR: Please ensure ELMC\_CONTRACT\_ADDRESS is set in your .env file.");

process.exit(1);

}

console.log("----------------------------------------------------");

console.log("🚨 Rescuing Stuck BNB via Defender Relayer");

console.log("ELMC Contract Address:", elmcAddress);

console.log("Amount to Rescue (BNB):", ethers.formatEther(amountToRescueBNB));

console.log("----------------------------------------------------");

// ELMC Coin ABI (only the rescueStuckBNB function)

const elmcAbi = [

"function rescueStuckBNB(uint256 amount) external",

"function owner() view returns (address)" // To check owner

];

const elmc = new ethers.Contract(elmcAddress, elmcAbi, signer);

// Get the owner of the ELMC contract (which should be the Gnosis Safe after transfer)

const currentOwner = await elmc.owner();

console.log("Current ELMC Contract Owner (Rescue Destination):", currentOwner);

// Check ELMC contract's BNB balance

const contractBNBBalance = await ethers.provider.getBalance(elmcAddress);

console.log("ELMC Contract's current BNB balance:", ethers.formatEther(contractBNBBalance));

if (contractBNBBalance < amountToRescueBNB) {

console.error(`ERROR: Insufficient BNB in contract. Requested: ${ethers.formatEther(amountToRescueBNB)}, Available: ${ethers.formatEther(contractBNBBalance)}`);

process.exit(1);

}

// Call rescueStuckBNB on the ELMCCoin contract

console.log(`Calling rescueStuckBNB on ELMCCoin contract to rescue ${ethers.formatEther(amountToRescueBNB)} BNB...`);

const rescueTx = await elmc.rescueStuckBNB(amountToRescueBNB);

console.log("Rescue TX sent:", rescueTx.hash);

await rescueTx.wait();

console.log(`✅ Successfully rescued ${ethers.formatEther(amountToRescueBNB)} BNB to ${currentOwner}.`);

console.log("----------------------------------------------------");

}

main().catch((error) => {

console.error(error);

process.exitCode = 1;

});