```
const hre = require("hardhat");
const fs = require("fs");

async function main() {
  const [deployer, payerVASP, payeeVASP] = await
hre.ethers.getSigners();

  console.log("Deploying contracts with account:",
deployer.address);
  console.log("Payer VASP:", payerVASP.address);
  console.log("Payee VASP:", payeeVASP.address);

  // 1. ✅ Deploy TestUSDC
  const TestUSDC = await hre.ethers.getContractFactory("TestUSDC");
  const usdc = await TestUSDC.deploy();
  await usdc.waitForDeployment();
  const usdcAddress = await usdc.getAddress();
  console.log("USDC deployed at:", usdcAddress);

  // 2. ✅ Deploy Payer CustodialWallet
  const CustodialWallet = await
hre.ethers.getContractFactory("CustodialWallet");
  const payerCustodialWallet = await
CustodialWallet.deploy(usdcAddress, payerVASP.address,
payerVASP.address);
  await payerCustodialWallet.waitForDeployment();
  const payerCustodialWalletAddress = await
payerCustodialWallet.getAddress();
  console.log("Payer CustodialWallet deployed at:",
payerCustodialWalletAddress);

  // ✅ Actually mint USDC to the wallet (not just log it)
  await usdc.mint(payerCustodialWalletAddress,
hre.ethers.parseUnits("2000", 18));
  console.log("✅ Minted 2000 USDC directly to Payer
CustodialWallet");

  // ✅ Direct minting (push model)
  await usdc.mint(payerCustodialWalletAddress,
hre.ethers.parseUnits("2000", 18));
  console.log("💰 Payer CustodialWallet directly initialized with
2000 USDC");

  // 3. ✅ Deploy Payee CustodialWallet
  const payeeCustodialWallet = await
CustodialWallet.deploy(usdcAddress, payeeVASP.address,
payeeVASP.address);
  await payeeCustodialWallet.waitForDeployment();
  const payeeCustodialWalletAddress = await
payeeCustodialWallet.getAddress();
  console.log("Payee CustodialWallet deployed at:",
payeeCustodialWalletAddress);
```

```
  // 4. ✅ Deploy Payer HTLCVault
  const HTLCVault = await
hre.ethers.getContractFactory("HTLCVault");
  const payerVault = await HTLCVault.deploy(usdcAddress);
  await payerVault.waitForDeployment();
  const payerVaultAddress = await payerVault.getAddress();
  console.log("Payer HTLCVault deployed at:", payerVaultAddress);

  // 5. ✅ Deploy Payee HTLCVault
  const payeeVault = await HTLCVault.deploy(usdcAddress);
  await payeeVault.waitForDeployment();
  const payeeVaultAddress = await payeeVault.getAddress();
  console.log("Payee HTLCVault deployed at:", payeeVaultAddress);

  // 6. ✅ Deploy LPToken
  const LPToken = await hre.ethers.getContractFactory("LPToken");
  const lpToken = await LPToken.deploy("Liquidity Provider Token",
"LPT", deployer.address);
  await lpToken.waitForDeployment();
  const lpTokenAddress = await lpToken.getAddress();
  console.log("LPToken deployed at:", lpTokenAddress);

  // 7. ✅ Deploy CrossChainPaymentProcessor
  const CrossChainPaymentProcessor = await
hre.ethers.getContractFactory("CrossChainPaymentProcessor");
  const processor = await CrossChainPaymentProcessor.deploy(
    payerVASP.address,
    payerCustodialWalletAddress,
    payerVaultAddress,
    usdcAddress
  );
  await processor.waitForDeployment();
  const processorAddress = await processor.getAddress();
  console.log("CrossChainPaymentProcessor deployed at:",
processorAddress);

  // ✅ Authorize processor
  await
payerCustodialWallet.connect(payerVASP).setAuthorizedProcessor(proce
ssorAddress, true);
  console.log("Authorized CrossChainPaymentProcessor in Payer
CustodialWallet");

  // ✅ Transfer LPToken ownership to processor
  await lpToken.transferOwnership(processorAddress);
  console.log("Transferred LPToken ownership to
CrossChainPaymentProcessor");

  // ✅ Save deployed addresses
  const deployedAddresses = {
    usdc: usdcAddress,
    payerCustodialWallet: payerCustodialWalletAddress,
    payeeCustodialWallet: payeeCustodialWalletAddress,
    payerVault: payerVaultAddress,
    payeeVault: payeeVaultAddress,
    lpToken: lpTokenAddress,
    processor: processorAddress,
    payerVASP: payerVASP.address,
    payeeVASP: payeeVASP.address
  };

  fs.writeFileSync("deployed_addresses.json",
JSON.stringify(deployedAddresses, null, 2));
  console.log("✅ All contract addresses saved to
deployed_addresses.json");
}

// Run
main().catch((error) => {
  console.error(error);
  process.exitCode = 1;
});
```