D4D4D4;--ch-t-background: #1E1E1E;--ch-t-lighter-inlineBackground: #1e1e1ee6;--ch-t-editor-background: #1E1E1E;--ch-t-editor-foreground: #D4D4D4;--ch-t-editor-rangeHighlightBackground: #ffffff0b;--ch-t-editor-infoForeground: #3794FF;--ch-t-editor-selectionBackground: #264F78;--ch-t-focusBorder: #007FD4;--ch-t-tab-activeBackground: #1E1E1E;--ch-t-tab-activeForeground: #ffffff;--ch-t-tab-inactiveBackground: #2D2D2D;--ch-t-tab-inactiveForeground: #ffffff80;--ch-t-tab-border: #252526;--ch-t-tab-activeBorder: #1E1E1E;--ch-t-editorGroup-border: #444444;--ch-t-editorGroupHeader-tabsBackground: #252526;--ch-t-editorLineNumber-foreground: #858585;--ch-t-input-background: #3C3C3C;--ch-t-input-foreground: #D4D4D4;--ch-t-icon-foreground: #C5C5C5;--ch-t-sideBar-background: #252526;--ch-t-sideBar-foreground: #D4D4D4;--ch-t-sideBar-border: #252526;--ch-t-list-activeSelectionBackground: #094771;--ch-t-list-activeSelectionForeground: #fffffe;--ch-t-list-hoverBackground: #2A2D2E; }

Multichain Safe Deployment

This guide will teach you how to replicate a Safe address across different chains using the Protocol Kit. This process includes initializing the Protocol Kit, configuring the Safes to deploy, predicting its address on different chains, and executing the deployment transactions.

For more detailed information, see theProtocol Kit Reference .

Prerequisites

- Node.js and npm(opens in a new tab)

Install dependencies

First, you need to install the Protocol Kit.

_10 pnpm add @safe-global/protocol-kit viem

Steps

Imports

Here are all the necessary imports for this guide.

_10 import Safe, { _10 PredictedSafeProps, _10 SafeAccountConfig, _10 SafeDeploymentConfig _10 } from '@safe-

global/protocol-kit' _10 import { waitForTransactionReceipt } from 'viem/actions' _10 import { gnosisChiado, sepolia } from 'viem/chains'

Create a signer

You need a signer to instantiate the Protocol Kit. This example uses a private key to obtain a signer, but other EIP-1193(opens in a new tab) compatible signers are also supported. For detailed information about signers, please refer to theProtocol Kit reference .

_10 const SIGNER_PRIVATE_KEY = // ...

Configure the Safe deployment

Define thepredictedSafe object with the configuration for all the Safe accounts you will deploy. Check the reference to learn about all the different configuration options.

_10 const safeAccountConfig: SafeAccountConfig = { _10 owners: ['0x...', '0x...', '0x...'], _10 threshold: 2 _10 // ... _10 } _10 _10 const predictedSafe: PredictedSafeProps = { _10 safeAccountConfig _10 // ... _10 }

Initialize the Protocol Kit

Initialize an instance of the Protocol Kit for each network where you want to deploy a new Safe smart account by calling theinit method. Pass theprovider with its corresponding value depending on the network, thesigner executing the deployment, and thepredictedSafe with the Safe account configuration.

_15 const protocolKitSepolia = await Safe.init({ _15 provider: sepolia.rpcUrls.default.http[0], _15 signer: SIGNER_PRIVATE_KEY, _15 predictedSafe, _15 onchainAnalytics // Optional _15 // ... _15 }) _15 _15 const protocolKitChiado = await Safe.init({ _15 provider: gnosisChiado.rpcUrls.default.http[0], _15 signer: PRIVATE_KEY, _15 predictedSafe, _15 onchainAnalytics // Optional _15 // ... _15 }) Optionally, you cantrack your Safe deployments and transactions on-chain by using theonchainAnalytics property.

Predict the Safe addresses

You can predict the Safe addresses by calling thegetAddress method from each Protocol Kit instance and ensure that the result addresses are the same.

_10 const sepoliaPredictedSafeAddress = await protocolKitSepolia.getAddress() _10 const chiadoPredictedSafeAddress = await protocolKitChiado.getAddress()

Deployment on Sepolia

Create the deployment transaction to deploy a new Safe account in Sepolia by calling thecreateSafeDeploymentTransaction method.

_10 const sepoliaDeploymentTransaction = _10 await protocolKitSepolia.createSafeDeploymentTransaction() Call thesendTransaction method from your Sepolia client instance and wait for the transaction to be executed.

_14 const sepoliaClient = _14 await protocolKitSepolia.getSafeProvider().getExternalSigner() _14 _14 const transactionHashSepolia = await sepoliaClient!.sendTransaction({ _14 to: sepoliaDeploymentTransaction.to, _14 value: BigInt(sepoliaDeploymentTransaction.value), _14 data: sepoliaDeploymentTransaction.data as 0x{string}, _14 chain: sepolia _14 }) _14 _14 await waitForTransactionReceipt(_14 sepoliaClient!, _14 { hash: transactionHashSepolia } _14) Once the deployment transaction is executed, connect the new Safe address to the Protocol Kit instance by calling theconnect method.

_10 const newProtocolKitSepolia = await protocolKitSepolia.connect({ _10 safeAddress: sepoliaPredictedSafeAddress _10 }) _10 _10 const isSepoliaSafeDeployed = await newProtocolKitSepolia.isSafeDeployed() // True _10 const sepoliaDeployedSafeAddress = await newProtocolKitSepolia.getAddress() If everything went well,isSepoliaSafeDeployed should betrue , and thesepoliaDeployedSafeAddress should equal thesepoliaPredictedSafeAddress .

Deployment on Chiado

Repeat the same steps to deploy a Safe with the same configuration on Chiado testnet.

_24 const chiadoDeploymentTransaction = _24 await protocolKitChiado.createSafeDeploymentTransaction() _24 _24 const chiadoClient = _24 await protocolKitChiado.getSafeProvider().getExternalSigner() _24 _24 const transactionHashChiado = await chiadoClient!.sendTransaction({ _24 to: chiadoDeploymentTransaction.to, _24 value: BigInt(chiadoDeploymentTransaction.value), _24 data: chiadoDeploymentTransaction.data as 0x{string}, _24 chain: gnosisChiado _24 }) _24 _24 await waitForTransactionReceipt(_24 chiadoClient!, _24 { hash: transactionHashChiado } _24) _24 _24 const newProtocolKitChiado = await protocolKitChiado.connect({ _24 safeAddress: chiadoPredictedSafeAddress

_24 }) _24 _24 const isChiadoSafeDeployed = await newProtocolKitChiado.isSafeDeployed() // True _24 const chiadoDeployedSafeAddress = await newProtocolKitChiado.getAddress() If everything went well,isChiadoSafeDeployed should betrue , and thechiadoDeployedSafeAddress should equal thechiadoPredictedSafeAddress .

In addition,chiadoDeployedSafeAddress andsepoliaDeployedSafeAddress should have the same value.

Recap and further reading

After following this guide, you are able to deploy multiple Safe accounts with the same address on different chains using the Protocol Kit.

Safe deployment Execute transactions Was this page helpful?

Report issue