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; }

Safe Deployment

This guide will teach you how to deploy a new Safe using the Protocol Kit. This process includes initializing the Protocol Kit, setting up your Safe configuration, and executing the deployment.

For more detailed information, see the Protocol Kit Reference .

Prerequisites

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

Install dependencies

First, you need to install some dependencies.

_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 { 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, butEIP-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 = // ...

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.

Optionally, you cantrack your Safe deployments and transactions on-chain by using theonchainAnalytics property.

_17 const safeAccountConfig: SafeAccountConfig = { _17 owners: ['0x...', '0x...', '0x...'], _17 threshold: 2 _17 // More optional properties _17 } _17 _17 const predictedSafe: PredictedSafeProps = { _17 safeAccountConfig _17 // More optional properties _17 } _17 _17 const protocolKit = await Safe.init({ _17 provider: sepolia.rpcUrls.default.http[0], _17 signer: SIGNER_PRIVATE_KEY, _17 predictedSafe, _17 onchainAnalytics // Optional _17 })

Predict the Safe address

You can predict the Safe address using thegetAddress method in the Protocol Kit.

_10 const safeAddress = await protocolKit.getAddress()

Create the deployment transaction

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

_10 const deploymentTransaction = await protocolKit.createSafeDeploymentTransaction()

Execute the deployment transaction

Once the deployment transaction object is ready, execute it using the provided signer or your preferred external Ethereum client.

_12 const client = await protocolKit.getSafeProvider().getExternalSigner() _12 _12 const transactionHash = await client.sendTransaction({ _12 to: deploymentTransaction.to, _12 value: BigInt(deploymentTransaction.value), _12 data: deploymentTransaction.data as 0x{string}, _12 chain: sepolia _12 }) _12 _12 const transactionReceipt = await client.waitForTransactionReceipt({ _12 hash: transactionHash _12 })

Reinitialize the Protocol Kit

Once the deployment transaction is executed, connect the new Safe address to the Protocol Kit instance by calling theconnect method.

_10 const newProtocolKit = await protocolKit.connect({ _10 safeAddress _10 }) _10 _10 const isSafeDeployed = await newProtocolKit.isSafeDeployed() // True _10 const safeAddress = await newProtocolKit.getAddress() _10 const safeOwners = await newProtocolKit.getOwners() _10 const safeThreshold = await newProtocolKit.getThreshold()

Recap and further reading

After following this guide, you are able to deploy new Safe smart accounts with the Protocol Kit.

Protocol Kit Multichain Safe deployment Was this page helpful?

Report issue