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; }

Execute transactions

In this quickstart guide, you will create a 2 of 3 multi-sig Safe and propose and execute a transaction to send some ETH out of this Safe.

To find more details and configuration options for available methods, see the Protocol Kit reference .

Prerequisites

1. Node.js and npm(opens in a new tab)
2. Three externally-owned accounts with Testnet ETH in at least one account

Install dependencies

First, we need to install some dependencies.

_10 yarn add @safe-global/protocol-kit \ _10 @safe-global/api-kit \ _10 @safe-global/safe-core-sdk-types

Initialize Signers and Providers

The signers trigger transactions to the Ethereum blockchain or off-chain transactions. The provider connects to the Ethereum blockchain.

You can get a public RPC URL from Chainlist(opens in a new tab) , however, public RPC URLs can be unreliable so you can also try a dedicated provider like Infura or Alchemy.

For this guide, we will be creating a Safe on the Sepolia Testnet.

_14 // https://chainlist.org/?search=sepolia&testnets=true _14 const RPC_URL = 'https://eth-sepolia.public.blastapi.io' _14 _14 // Initialize signers _14 const OWNER_1_ADDRESS = // ... _14 const OWNER_1_PRIVATE_KEY = // ... _14 _14 const OWNER_2_ADDRESS = // ... _14 const OWNER_2_PRIVATE_KEY = // ... _14 _14 const OWNER_3_ADDRESS = // ... _14 _14 const provider = new ethers.JsonRpcProvider(RPC_URL) _14 const owner1Signer = new ethers.Wallet(OWNER_1_PRIVATE_KEY, provider)

Initialize the API Kit

TheAPI Kit(opens in a new tab) consumes theSafe Transaction Service API(opens in a new tab) . To use this library, create a new instance of theSafeApiKit class, imported from@safe-global/api-kit . In chains where Safe provides a Transaction Service, it's enough to specify thechainId. You can specify your own service using the optionaltxServiceUrl parameter.

You will be using Sepolia for this tutorial, however, you can also getservice URLs for different networks .

_11 import SafeApiKit from '@safe-global/api-kit' _11 _11 const apiKit = new SafeApiKit({ _11 chainId: 1n _11 }) _11 _11 // Or using a custom service _11 const apiKit = new SafeApiKit({ _11 chainId: 1n, // set the correct chainId _11 txServiceUrl: 'https://url-to-your-custom-service' _11 })

Initialize the Protocol Kit

TheSafeFactory class allows the deployment of new Safe accounts while theSafe class represents an instance of a specific one.

_10 import { SafeFactory } from '@safe-global/protocol-kit' _10 _10 const safeFactory = await SafeFactory.init({ _10 provider: RPC_URL, _10 signer: OWNER_1_PRIVATE_KEY _10 })

Deploy a Safe

Calling thedeploySafe method will deploy the desired Safe and return a Protocol Kit initialized instance ready to be used. Check themethod reference for more details on additional configuration parameters and callbacks.

_20 import { SafeAccountConfig } from '@safe-global/protocol-kit' _20 _20 const safeAccountConfig: SafeAccountConfig = { _20 owners: [_20 await OWNER_1_ADDRESS, _20 await OWNER_2_ADDRESS, _20 await OWNER_3_ADDRESS _20], _20 threshold: 2, _20 // Optional params _20 } _20 _20 / *This Safe is tied to owner 1 because the factory was initialized with the owner 1 as the signer.* / _20 const protocolKitOwner1 = await safeFactory.deploySafe({ safeAccountConfig }) _20 _20 const safeAddress = await protocolKitOwner1.getAddress() _20 _20 console.log('Your Safe has been deployed:') _20 console.log(https://sepolia.etherscan.io/address{safeAddress}) _20 console.log(https://app.safe.global/sep:{safeAddress})

Send ETH to the Safe

You will send some ETH to this Safe.

_13 const safeAddress = protocolKit.getAddress() _13 _13 const safeAmount = ethers.parseUnits('0.01', 'ether').toHexString() _13 _13 const transactionParameters = { _13 to: safeAddress, _13 value: safeAmount _13 } _13 _13 const tx = await owner1Signer.sendTransaction(transactionParameters) _13 _13 console.log('Fundraising.') _13 console.log(Deposit Transaction: https://sepolia.etherscan.io/tx{tx.hash})

Making a transaction from a Safe

The first signer will sign and propose a transaction to send 0.005 ETH out of the Safe. Then, the second signer will add their own proposal and execute the transaction since it meets the 2 of 3 thresholds.

At a high level, making a transaction from the Safe requires the following steps:

Overview

The high-level overview of a multi-sig transaction is PCE: Propose. Confirm. Execute.

1. First signer proposes a transaction
2.
    1. Create transaction: define the amount, destination, and any additional data
3.
    1. Perform an off-chain signature of the transaction before proposing
4.
    1. Submit the transaction and signature to the Safe Transaction Service
5. Second signer confirms the transaction
6.

1. Get pending transactions from the Safe service
7. 1. Perform an off-chain signature of the transaction
8. 1. Submit the signature to the service
9. Anyone executes the transaction
10. 1. In this example, the first signer executes the transaction
11. 1. Anyone can get the pending transaction from the Safe service
12. 1. Account executing the transaction pays the gas fee

Create a transaction

For more details on what to include in a transaction seecreateTransaction method.

_13 import { MetaTransactionData } from '@safe-global/safe-core-sdk-types' _13 _13 // Any address can be used. In this example you will use vitalik.eth _13 const destination = '0xd8dA6BF26964aF9D7eEd9e03E53415D37aA96045' _13 const amount = ethers.parseUnits('0.005', 'ether').toString() _13 _13 const safeTransactionData: MetaTransactionData = { _13 to: destination, _13 data: '0x', _13 value: amount _13 } _13 // Create a Safe transaction with the provided parameters _13 const safeTransaction = await protocolKitOwner1.createTransaction({ transactions: [safeTransactionData] })

Track the Safe transaction

Optionally, you can track all your Safe transactions on-chain by attaching an on-chain identifier to thedata property.

This identifier must be unique for every project and has a length of 16 bytes. You can create a random one or derive it from a text string, maybe from your project name:

_10 const onchainIdentifier = toHex(_10 'TEXT_TO_DERIVE_THE_IDENTIFIER', // It could be your project name _10 { size: 16 } _10) Once generated, fill theEcosystem On-chain Tracking Form(opens in a new tab)and provide the value of youronchainIdentifier .

Add theonchainIdentifier at the end of the Safe transactiondata .

_10 safeTransaction.data.data = concat([_10 safeOperation.data.data as 0x{string}, _10 onchainIdentifier _10]).toString()

Propose the transaction

To propose a transaction to the Safe Transaction Service we need to call theproposeTransaction method from the API Kit instance.

For a full list and description of the properties seeproposeTransaction in the API Kit reference.

_13 // Deterministic hash based on transaction parameters _13 const safeTxHash = await protocolKitOwner1.getTransactionHash(safeTransaction) _13 _13 // Sign transaction to verify that the transaction is coming from owner 1 _13 const senderSignature = await protocolKitOwner1.signHash(safeTxHash) _13 _13 await apiKit.proposeTransaction({ _13 safeAddress, _13 safeTransactionData: safeTransaction.data, _13 safeTxHash, _13 senderAddress: OWNER_1_ADDRESS, _13 senderSignature: senderSignature.data _13 })

Get pending transactions

_10 const pendingTransactions = (await apiKit.getPendingTransactions(safeAddress)).results

Confirm the transaction: Second confirmation

When owner 2 is connected to the application, the Protocol Kit should be initialized again with the existing Safe address the address of the owner 2 instead of the owner 1.

_12 // Assumes that the first pending transaction is the transaction you want to confirm _12 const transaction = pendingTransactions[0] _12 const safeTxHash = transaction.safeTxHash _12 _12 const protocolKitOwner2 = await Safe.init({ _12 provider: RPC_URL, _12 signer: OWNER_2_PRIVATE_KEY, _12 safeAddress _12 }) _12 _12 const signature = await protocolKitOwner2.signHash(safeTxHash) _12 const response = await apiKit.confirmTransaction(safeTxHash, signature.data)

Execute the transaction

Anyone can execute the Safe transaction once it has the required number of signatures. In this example, owner 1 will

execute the transaction and pay for the gas fees.

_10 const safeTransaction = await apiKit.getTransaction(safeTxHash) _10 const executeTxResponse = await protocolKit.executeTransaction(safeTransaction) _10 const receipt = await executeTxResponse.transactionResponse?.wait() _10 _10 console.log('Transaction executed:') _10 console.log(https://sepolia.etherscan.io/tx{receipt.transactionHash})

Confirm that the transaction was executed

You know that the transaction was executed if the balance in your Safe changes.

_10 const afterBalance = await protocolKit.getBalance() _10 _10 console.log(The final balance of the Safe: {ethers.formatUnits(afterBalance, 'ether')} ETH) _10 node index.js _10 _10 Fundraising. _10 _10 Initial balance of Safe: 0.01 ETH _10 Buying a car. _10 The final balance of the Safe: 0.005 ETH

Conclusion

In this quickstart, you learned how to create and deploy a new Safe account and to propose and then execute a transaction from it.

Protocol Kit Signatures Was this page helpful?

Report issue