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; }

Propose and confirm transactions

In this guide you will learn how to propose transactions to the service and collect the signatures from the owners so they become executable.

For more detailed information, see theAPI Kit Reference .

Prerequisites

1. Node.js and npm(opens in a new tab)
2. A Safe with several signers

Steps

Install dependencies

First, you need to install some dependencies.

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

Imports

Here are all the necessary imports for this guide.

_10 import SafeApiKit from '@safe-global/api-kit' _10 import Safe from '@safe-global/protocol-kit' _10 import { _10

MetaTransactionData, _10 OperationType _10 } from '@safe-global/types-types'

Setup

We will use a Safe account setup with two or more signers, and threshold two, so at least multiple signatures will need to be collected when executing a transaction.

_10 // https://chainlist.org/?search=sepolia&testnets=true _10 const RPC_URL = 'https://eth-sepolia.public.blastapi.io' _10 _10 const SAFE_ADDRESS = // ... _10 _10 const OWNER_1_ADDRESS = // ... _10 const OWNER_1_PRIVATE_KEY = // ... _10 _10 const OWNER_2_PRIVATE_KEY = // ...

Initialize the API Kit

Firstly, you need to create an instance of the API Kit. In chains where the Safe Transaction Service is supported, it's enough to specify thechainId property.

_10 const apiKit = new SafeApiKit({ _10 chainId: 1n _10 }) Alternatively, you can use a custom service using the optionaltxServiceUrl property.

_10 const apiKit = new SafeApiKit({ _10 chainId: 1n, // set the correct chainId _10 txServiceUrl: 'https://url-to-your-custom-service' _10 })

Initialize the Protocol Kit

To handle transactions and signatures, you need to create an instance of the Protocol Kit with theprovider ,signer andsafeAddress .

_10 const protocolKitOwner1 = await Safe.init({ _10 provider: RPC_URL, _10 signer: OWNER_1_PRIVATE_KEY, _10 safeAddress: SAFE_ADDRESS _10 })

Propose a transaction to the service

Before a transaction can be executed, any of the Safe signers needs to initiate the process by creating a proposal of a transaction. This transaction is sent to the service to make it accessible by the other owners so they can give their approval and sign the transaction as well.

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

_23 // Create transaction _23 const safeTransactionData: MetaTransactionData = { _23 to: '0x', _23 value: '1', // 1 wei _23 data: '0x', _23 operation: OperationType.Call _23 } _23 _23 const safeTransaction = await protocolKitOwner1.createTransaction({ _23 transactions: [safeTransactionData] _23 }) _23 _23 const safeTxHash = await protocolKitOwner1.getTransactionHash(safeTransaction) _23 const signature = await protocolKitOwner1.signHash(safeTxHash) _23 _23 // Propose transaction to the service _23 await apiKit.proposeTransaction({ _23 safeAddress: SAFE_ADDRESS, _23 safeTransactionData: safeTransaction.data, _23 safeTxHash, _23 senderAddress: OWNER_1_ADDRESS, _23 senderSignature: signature.data _23 })

Retrieve the pending transactions

Different methods in the API Kit are available to retrieve pending transactions depending on the situation. To retrieve a transaction given the Safe transaction hash use the method that's not commented.

_10 const transaction = await service.getTransaction(safeTxHash) _10 // const transactions = await service.getPendingTransactions() _10 // const transactions = await service.getIncomingTransactions() _10 // const transactions = await service.getMultisigTransactions() _10 // const transactions = await service.getModuleTransactions() _10 // const transactions = await service.getAllTransactions()

Confirm the transaction

In this step you need to sign the transaction with the Protocol Kit and submit the signature to the Safe Transaction Service using theconfirmTransaction method.

_14 const protocolKitOwner2 = await Safe.init({ _14 provider: RPC_URL, _14 signer: OWNER_2_PRIVATE_KEY, _14 safeAddress: SAFE_ADDRESS _14 }) _14 _14 const safeTxHash = transaction.transactionHash _14 const signature = await protocolKitOwner2.signHash(safeTxHash) _14 _14 // Confirm the Safe transaction _14 const signatureResponse = await apiKit.confirmTransaction(_14 safeTxHash, _14 signature.data _14) The Safe transaction is now ready to be executed. This can be done using theSafe{Wallet} web(opens in a new tab) interface, theProtocol Kit , theSafe CLI or any other tool that's available.

Report issue