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; }

Transactions with off-chain signatures

This guide shows how to interact with the Safe Transaction Service API to create, sign, and execute transactions with the owners of a Safe account.

The different steps are implemented usingCurl(opens in a new tab) requests, theSafe{Core} SDK(opens in a new tab) TypeScript library and thesafe-eth-py(opens in a new tab) Python library.

Prerequisites

1. Node.js and npm(opens in a new tab)
2. when using the Safe{Core} SDK.
3. Python(opens in a new tab)
4. = 3.9 when usingsafe-eth-py
5. .
6. Have a Safe account configured with a threshold of 2, where two signatures are needed.

Steps

Install dependencies

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

Imports

TypeScript Python _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/safe-core-sdk-types'

Create a Safe transaction

TypeScript Python Curl _18 // Initialize the Protocol Kit with Owner A _18 const protocolKitOwnerA = await Safe.init({ _18 provider: config.RPC_URL, _18 signer: config.OWNER_A_PRIVATE_KEY, _18 safeAddress: config.SAFE_ADDRESS _18 }) _18 _18 // Create a Safe transaction _18 const safeTransactionData: MetaTransactionData = { _18 to: config.TO, _18 value: config.VALUE, _18 data: '0x', _18 operation: OperationType.Call _18 } _18 _18 const safeTransaction = await protocolKitOwnerA.createTransaction({ _18 transactions: [safeTransactionData] _18 })

Sign the transaction

TypeScript Python Curl _10 // Sign the transaction with Owner A _10 const safeTxHash = await protocolKitOwnerA.getTransactionHash(safeTransaction) _10 const signatureOwnerA = await protocolKitOwnerA.signHash(safeTxHash)

Send the transaction to the service

TypeScript Python Curl _13 // Initialize the API Kit _13 const apiKit = new SafeApiKit({ _13 chainId: 11155111n _13 }) _13 _13 // Send the transaction to the Transaction Service with the signature from Owner A _13 await apiKit.proposeTransaction({ _13 safeAddress: config.SAFE_ADDRESS, _13 safeTransactionData: safeTransaction.data, _13 safeTxHash, _13 senderAddress: config.OWNER_A_ADDRESS, _13 senderSignature: signatureOwnerA.data _13 })

Collect missing signatures

Get the pending transaction

TypeScript Python Curl _10 const signedTransaction = await apiKit.getTransaction(safeTxHash)

Add missing signatures

TypeScript Python Curl _15 // Initialize the Protocol Kit with Owner B _15 const protocolKitOwnerB = await Safe.init({ _15 provider: config.RPC_URL, _15 signer: config.OWNER_B_PRIVATE_KEY, _15 safeAddress: config.SAFE_ADDRESS _15 }) _15 _15 // Sign the transaction with Owner B _15 const signatureOwnerB = await protocolKitOwnerB.signHash(safeTxHash) _15 _15 // Send the transaction to the Transaction Service with the signature from Owner B _15 await apiKit.confirmTransaction(_15 safeTxHash, _15 signatureOwnerB.data _15)

Execute the transaction

TypeScript Python _10 const transactionResponse = _10 await protocolKitOwnerA.executeTransaction(signedTransaction)

Get the executed transaction

TypeScript Python Curl _10 const transactions = await apiKit.getMultisigTransactions(config.SAFE_ADDRESS) _10 _10 if (transactions.results.length > 0) { _10 console.log('Last executed transaction', transactions.results[0]) _10 }

Supported Networks Messages