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; }

Messages with off-chain signatures

This guide shows how to interact with the Safe Transaction Service API to create and sign messages with a Safe account.

The different steps are implemented using Curl(opens in a new tab) requests, the Safe{Core} SDK(opens in a new tab) TypeScript library and the safe-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 using safe-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, { AddMessageProps } from '@safe-global/api-kit' _10 import Safe, { hashSafeMessage } from '@safe-global/protocol-kit'

Create a Safe message

TypeScript Python Curl _11 // Initialize the Protocol Kit with Owner A _11 const protocolKitOwnerA = await Safe.init({ _11 provider: config.RPC_URL, _11 signer: config.OWNER_A_PRIVATE_KEY, _11 safeAddress: config.SAFE_ADDRESS _11 }) _11 _11 const rawMessage: string = 'A Safe Message - ' + Date.now() _11 _11 // Create a Safe message _11 const safeMessage = protocolKitOwnerA.createMessage(rawMessage)

Sign the message

TypeScript Python Curl _10 // Sign the message with Owner A _10 const signedMessageOwnerA = await protocolKitOwnerA.signMessage(safeMessage)

Send the message to the service

TypeScript Python Curl _12 // Initialize the API Kit _12 const apiKit = new SafeApiKit({ _12 chainId: 11155111n _12 }) _12 _12 const messageProps: AddMessageProps = { _12 message: rawMessage, _12 signature: signedMessageOwnerA.encodedSignatures() _12 } _12 _12 // Send the message to the Transaction Service with the signature from Owner A _12 apiKit.addMessage(config.SAFE_ADDRESS, messageProps)

Collect the missing signatures

Get the pending message

TypeScript Python Curl _14 // Initialize the Protocol Kit with Owner B _14 const protocolKitOwnerB = await Safe.init({ _14 provider: config.RPC_URL, _14 signer: config.OWNER_B_PRIVATE_KEY, _14 safeAddress: config.SAFE_ADDRESS _14 }) _14 _14 // Get the Safe message hash _14 const safeMessageHash = await protocolKitOwnerB.getSafeMessageHash(_14 hashSafeMessage(rawMessage) _14) _14 _14 // Get the Safe message _14 const safeServiceMessage = await apiKit.getMessage(safeMessageHash)

Add missing signatures

TypeScript Python Curl _11 // Sign the message with Owner B _11 const signedMessageOwnerB = await protocolKitOwnerB.signMessage(safeServiceMessage) _11 _11 // Get Owner B address _11 const ownerBAddress = '0x...' _11 _11 // Send the message to the Transaction Service with the signature from Owner B _11 await apiKit.addMessageSignature(_11 safeMessageHash, _11 signedMessageOwnerB.getSignature(ownerBAddress)?.data || '0x' _11)

Transactions Delegates

Was this page helpful?

Report issue