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; }

Onramp with Monerium

Monerium(opens in a new tab) provides EURe, a regulated stablecoin on Ethereum, Polygon, and Gnosis. This guide demonstrates how to use theMonerium SDK(opens in a new tab) and the Safe{Core} SDK together to enable direct transfers from Safe accounts to an IBAN via the SEPA network and vice versa.

Prerequisites

1. Node.js and npm(opens in a new tab)
2. Monerium account and application(opens in a new tab)
3. A web application using your favorite CLI and language. For exampleReact with NextJS(opens in a new tab)
4. ,Vue with Nuxt(opens in a new tab)
5. orSvelte with SvelteKit(opens in a new tab)
6. .
7. Adeployed Safe
8. for your users.

Overview

The main steps of this tutorial are:

1. Authentication with Monerium and Safe: This requires signing a message with the Safe and using the Monerium Authentication flow.
2. Sending an order from the Safe to an IBAN involves sending an order to Monerium and signing the order with the Safe.

Step 1: Authenticate with Monerium and Safe

Sign the link message with the Safe

First, your users have to sign a message with the Safe to prove ownership of the Safe. Monerium will scan the signed messages in the Safe to verify ownership.

Use the Safe{Core} SDK, as shown below, to sign a message with multiple owners (for example, a two-out-of-three Safe). This example programmatically signs a message with two owners. In practice, one owner proposes the transaction, and another owner confirms it using theSafe{Wallet}(opens in a new tab) UI.

_52 import Safe, { _52 getSignMessageLibContract, _52 hashSafeMessage _52 } from '@safe-global/protocol-kit' _52 import { constants } from '@monerium/sdk' _52 _52 // Initialize the Safe{Core} SDK and link it to an existing Safe _52 const protocolKit = await Safe.init({ _52 provider: RPC_URL, // set a valid RPC URL _52 signer: OWNER_1_PRIVATE_KEY, // set the private key of the first Safe owner _52 safeAddress _52 }) _52 _52 // Create a signed message by creating a transaction to the signMessage contract _52 const signMessageContract = await getSignMessageLibContract({ _52 safeProvider: protocolKit.getSafeProvider(), _52 safeVersion: await protocolKit.getContractVersion() _52 }) _52 _52 // Let the contract encode the message's hash to get the transaction data _52 const txData = signMessageContract.encode('signMessage', [_52 hashSafeMessage(constants.LINK_MESSAGE) // 'I hereby declare that I am the address owner.' _52]) _52 _52 // Assemble a transaction object _52 const safeTransactionData = { _52 to: await signMessageContract.getAddress(), _52 value: '0', _52 data: txData, _52 operation: OperationType.DelegateCall _52 } _52 _52 // Create a transaction _52 const signMessageTx = await protocolKit.createTransaction({ _52 transactions: [safeTransactionData] _52 }) _52 _52 // Sign the transaction with the first owner _52 const signedTx = await protocolKit.signTransaction(signMessageTx) _52 _52 // Connect the protocol kit to the second owner _52 const protocolKitOfOwner2 = await protocolKit.connect({ _52 signer: OWNER_2_PRIVATE_KEY _52 }) _52 _52 // Sign and execute the transaction as the second owner _52 const transactionResult = await protocolKitOfOwner2.executeTransaction(_52 signedTx _52) _52 _52 // Check the transaction hash to see if the transaction settled _52 console.log('transactionResult', transactionResult) TheprotocolKit is an instance of theSafe class. For more information on instantiating the Protocol Kit, refer to theProtocol Kit Quickstart section .

Initialize the Monerium client and authenticate the users

After the message is signed and the transaction is executed, the users can authenticate with Monerium.

_15 import { MoneriumClient } from '@monerium/sdk' _15 _15 // Initialize the Monerium Client _15 const monerium = new MoneriumClient({ _15 clientId: 'a1b2c3-x7y8y9', // Get your client ID from Monerium _15 environment: 'sandbox' // Use the appropriate Monerium environment ('sandbox' | 'production') _15 }) _15 _15 // Start the Monerium authentication flow and send the users to Monerium _15 await monerium.authorize({ _15 address: safeAddress, // The address of the users' Safe _15 signature: '0x', // '0x' for Safe authentication lets Monerium look for the signature on-chain _15 redirectUrl: 'http://localhost:3000/return', // URL where Monerium will redirect the users after authenticating _15 chainId: 11155111 // Chain ID of Sepolia in this example _15 }) Callingauthorize will redirect the users to the Monerium login page.

Authenticate with Monerium

At Monerium, the users need to log in or create an account. Once logged in, Monerium will verify the ownership of the Safe by checking the signed message. After successful verification, Monerium will create an IBAN and link it to the Safe. The users will then be redirected to the specifiedredirectUrl with the new session id as a GET parameter.

Finish the authentication

Once the users land back on your page, finish the authentication process.

_10 // Returns true, if the users are authorized _10 const isAuthorized = await monerium.getAccess() Congratulations, you authenticated your users with Monerium and linked the Safe to the Monerium account.

Step 2: Place an Order

Once users are authenticated with Monerium, they can place an order to transfer tokens from their Safe to an IBAN.

Get some tokens

First, your users needs to obtain some EURe test tokens on Sepolia:

- Log into theMonerium Sandbox(opens in a new tab)
- account.
- Click on theADD MONEY
- button.
- Create a test IBAN transfer onto the account. The tokens from this test transfer will be available in the Safe.

Send an order to Monerium

To send tokens from a Safe to an IBAN, users must send the order to Monerium and sign a message with their account.

_30 import { placeOrderMessage } from '@monerium/sdk' _30 _30 const amount = '10' // Specify the amount in Euro _30 const iban = 'DK4878805291075472' // The target IBAN _30 _30 // 'Send EUR 10 to DK4878805291075472 at Fri, 17 May 2024 20:55:29Z' _30 const orderMessage = placeOrderMessage(amount, 'eur', iban) _30 _30 // Send the order to the Monerium backend _30 const order = await moneriumClient.placeOrder({ _30 amount, _30 signature: '0x', _30 currency: 'eur', _30 address: safeAddress, // the Safe address _30 counterpart: { _30 identifier: { _30 standard: 'iban', _30 iban _30 }, _30 details: { _30 firstName: 'User', _30 lastName: 'Userson', _30 county: 'AL' _30 } _30 }, _30 message: orderMessage, _30 memo: 'Powered by Monerium SDK', _30 chain: 'ethereum', _30 network: 'sepolia' _30 }) Monerium will listen for the sign message transaction of the following step on the selected chain and execute the order once the transaction settles.

Sign the order with the Safe

Now, the Safe needs to sign the order. To do this, send a sign message transaction to the blockchain. In practice, one owner might propose this transaction using the Safe Transaction Service. The other owner confirms the message and sends the transaction through theSafe{Wallet}(opens in a new tab) UI.

_26 // Hash and encode the order message _26 const txData = signMessageContract.encode('signMessage', [_26 hashSafeMessage(orderMessage) _26]) _26 _26 // Assemble a transaction object _26 const safeTransactionData = { _26 to: await signMessageContract.getAddress(), _26 value: '0', _26 data: txData, _26 operation: OperationType.DelegateCall _26 } _26 _26 // Create a transaction with the Safe{Core} SDK _26 const signMessageTx = await protocolKit.createTransaction({ _26 transactions: [safeTransactionData] _26 }) _26 _26 // Sign the transaction with the first owner _26 const signedTx = await protocolKit.signTransaction(signMessageTx) _26 _26 // Sign and execute the transaction with the second owner _26 const transactionResult = await protocolKitOfOwner2.executeTransaction(signedTx) _26 _26 // Verify on-chain settlement _26 console.log('transactionResult', transactionResult) Monerium will execute the order once the transaction settles. Well done! You have linked your users' Safes to an IBAN, bridging the gap between blockchain and traditional payment rails.

Further reading

- Monerium Developer Portal(opens in a new tab)
- Addevent listeners(opens in a new tab)
- to make your app more interactive.

Onramp Stripe Was this page helpful?

Report issue