SafeAuth Kit SDK

Safe is a platform that makes it possible for anyone to own a digital account. It establishes universal and transparent standards for the storage of digital assets, data, and identity. WithSafeAuth Kit , you can easily create smart wallets that work seamlessly withERC-4337 , using the user's EOA as the smart wallet's signer. This simplifies the process of incorporatingAccount Abstraction features into your application.

tip Check out ourexample app andrepository for an end-to-endWeb3Auth andSafeAuth Kit integration .

Install SafeAuth Kitâ

To get started, install the@safe-global/auth-kit SDK from Safe.

npm i @safe-global/auth-kit

Create a SafeAuthPack instanceâ

To integrate SafeAuth into your application, start by creating an instance of theSafeAuthPack class from the@safe-global/auth-kit package. This instance will be the entry point for managing authentication and transactions.

import

{ SafeAuthPack , SafeAuthConfig , SafeAuthInitOptions }

from

"@safe-global/auth-kit" ;

const safeAuthInitOptions : SafeAuthInitOptions =

{ showWidgetButton :

false ,

// Set to true to show the SafeAuth widget button chainConfig :

{ blockExplorerUrl :

"https://sepolia.etherscan.io" ,

// The block explorer URL chainId :

"0xaa36a7" ,

// The chain ID displayName :

"Ethereum Sepolia" ,

// The chain name rpcTarget :

"https://rpc.ankr.com/eth_sepolia" ,

// The RPC target ticker :

"ETH" ,

// The chain ticker tickerName :

"Ethereum" ,

// The chain ticker name } , } ;

const safeAuthPack =

new

SafeAuthPack ( ) ; await safeAuthPack . init (safeAuthInitOptions) ; You should always call theinit() method afterward before interacting with the pack. The init method initializes the provided Web3Auth SDK and Safe services. It creates an embedded browser wallet within an iframe, establishing communication through the internally generated EIP-1193 provider.

Sign in to an Ethereum accountâ

Sign Inâ

To start the authentication process, simply call thesignIn() method. This method is usually triggered when a user clicks on aSign In button on your web application. Once the user has successfully signed in, a new Ethereum Wallet will be created. This wallet will be utilized for all future logins and can beshared across different applications .

const safeAuthSignInResponse =

await safeAuthPack . signIn ( ) ; The returned safeAuthSignInResponse contains the user's Ethereum address and associated Safe addresses.

This method retrieves Safe addresses owned by the EOA, but does not create a Safe. AuthKitSignInData { eoa :

string

// The safe signer safes ? :

string [ ]

// The list of associated Safe addresses in the chain }

Create Safeâ

Creation of a Safe is done by using the SafeFactory from theProtocol Kit . The Protocol Kit is a library that provides a set of tools to interact with the Safe smart contract. To create a Safe, use the following code:

import

{ ethers , BrowserProvider , Eip1193Provider }

from

"ethers" ; import

{ EthersAdapter , SafeFactory }

from

"@safe-global/protocol-kit" ;

const provider =

new

BrowserProvider (safeAuthPack ?. getProvider ( )

as Eip1193Provider) ; const signer =

await provider . getSigner ( ) ; const ethAdapter =

new

EthersAdapter ({ ethers , signerOrProvider : signer , }

as

any) ;

const safeFactory =

await SafeFactory . create ({ ethAdapter }) ; const safe =

await safeFactory . deploySafe ({ safeAccountConfig :

{ threshold :

1 , owners :

[safeAuthSignInResponse ?. eoa as

string]

} , }) ; console . log ("SAFE Created!" ,

await safe . getAddress ( )) ;

Providerâ

After authentication, usegetProvider() to obtain the Ethereum provider instance, compatible withEIP-1193 .

safeAuthPack . getProvider ( ) ;

Sign Outâ

To sign out the user, use thesignOut() method.

await safeAuthPack . signOut ( ) ;

Signing and executing transactionsâ

Combine SafeAuth Kit with the Protocol Kit to connect to a Safe and perform transactions.

import

{ ethers , BrowserProvider , Eip1193Provider }

from

"ethers" ; import

{ EthersAdapter }

from

"@safe-global/protocol-kit" ;

const provider =

new

BrowserProvider (safeAuthPack ?. getProvider ( )

as Eip1193Provider) ; const signer =

await provider . getSigner ( ) ;

// Create the Safe EthersAdapter const ethAdapter =

new

EthersAdapter ({ ethers , signerOrProvider : signer || provider , }) ;

const safeAddress = safeAuthSignInResponse ?. safes ?. [0]

||

"0x" ;

// Instantiate the protocolKit const protocolKit =

await Safe . create ({ ethAdapter , safeAddress , }) ;

// Create a Safe transaction with the provided parameters const safeTransactionData : MetaTransactionData =

{ to : ethers . getAddress (safeAuthSignInResponse ?. eoa ||

"0x") , data :

"0x" , value : ethers . parseUnits ("0.0001" ,

"ether") . toString ( ) , } ;

const safeTransaction =

await protocolKit . createTransaction ({ safeTransactionData , }) ;

// Sign transaction const tx =

await protocolKit . signTransaction (safeTransaction) ;

// Execute the transaction const txResult =

await protocolKit . executeTransaction (tx) ;

Sign messages using theSafeAuthPack

â

Sign messages or transactions using your preferred web3 library.

- web3.js
- ethers.js

// Using web3 const web3 =

new

Web3 (safeAuthPack . getProvider ( )) ;

const message =

"Safe meets Web3Auth" ; const address = safeAuthSignInResponse ?. eoa ||

"0x" ; await web3 . eth . personal . sign (message , address) ; await web3 . eth . signTransaction (tx) ; await web3 . eth . sendTransaction (tx) ; // Using ethers const provider =

new

BrowserProvider (safeAuthPack ?. getProvider ( )

as Eip1193Provider) ; const signer =

await provider . getSigner ( ) ;

await signer . signMessage (message) ;

await signer . signTransaction (tx) ; await signer . sendTransaction (tx) ; The integration makes use of Web3Auth Wallet Services to offer users a secure and hassle-free authentication experience. Edit this page Previous Share Transfer Module Next Wallet Services