MoneriumPack

Monerium Pack enables using Safe withMonerium(opens in a new tab) , a regulated platform that facilitates the use of e-money tokens on the blockchain.

Install dependencies

To use theMoneriumPack , you need to install the Monerium SDK in addition to the@safe-global/onramp-kit package.

yarn

add

@safe-global/onramp-kit

@safe-global/protocol-kit

@monerium/sdk

Reference

TheMoneriumPack class enables the use of Monerium services with Safe. To use it, create an instance of the pack and pass it to theSafeOnrampKit init method.

This pack allows you to "Login with Monerium" by creating a connection between your Safe address and your Monerium account. This pack starts an authentication flow that uses the Monerium SDK to gain access to your account.

const

moneriumPack

=

new

MoneriumPack ({ clientId :

'YOUR_CLIENT_ID' , redirectUrl :

'URL_AFTER_AUTHENTICATION' environment: 'sandbox' }) await

moneriumPack .init (moneriumInitOptions)

new MoneriumPack(moneriumConfig)

Parameters

- moneriumConfig
-
    - The configuration for the Monerium pack. The options are:

MoneriumProviderConfig { clientId : string redirectUrl : string environment :

'production'

|

'sandbox' }

TheclientId is the secret representing the "Authorization Code Flow" for your Monerium account. To get yourclientId :

1. Log in toyour account(opens in a new tab)
2. .
3. Create anew application(opens in a new tab)
4. .

TheredirectUrl is an URI that will be used to send the user back to the app after they complete the authentication process.

Theenvironment is the environment for the Monerium SDK. You can choose betweenproduction andsandbox .

Theproduction environment will use the production Monerium services and the accounts will need to go through a KYC process. Real money will be transferred. The sandbox environment will use the Moneriumsandbox services(opens in a new tab) and no KYC is required. Fake money will be used.

Caveats You should always call theinit() method afterwards before interacting with the pack.

init(moneriumInitOptions)

Theinit method initializes the Monerium SDK and the Safe services by creating a new instance of theSafeMoneriumClient (opens in a new tab) class. This class extends theMoneriumClient (opens in a new tab) class from the Monerium SDK and adds extra features to use it with the Safe services.

Parameters

TheMoneriumInitOptions options to be passed to theinit method are:

MoneriumInitOptions { protocolKit : Safe }

- protocolKit
-
    - To use theMoneriumPack
- , you need to add Protocol Kit as a dependency for your project and create an instance of theSafe (opens in a new tab)
- class.

open(moneriumOpenOptions)

Theopen() method initiates the authentication process with Monerium. It opens a popup window with the Monerium authentication page.

Parameters

TheMoneriumOpenOptions options to be passed to theopen method are:

MoneriumOpenOptions { initiateAuthFlow ?: boolean }

- initiateAuthFlow
-
    - This flag should be added the first timeopen()
- is called to prompt the user to the authorization flow. Once authenticated, the dApp can call again theopen()
- method to get theMoneriumClient
- .

Take a look tothe example(opens in a new tab) for more information.

Returns

ASafeMoneriumClient instance. This instance contains all methods and properties from the Monerium SDK, plus some extra ones for using the Safe services. You can use them in your application to create any flow you need.

For more information about the available methods, refer to the Monerium SDKdocumentation(opens in a new tab) .

The Monerium SDK will be enhanced with Safe-related methods, mainly used in theMoneriumPack so you won't need to call them directly. The exception will be thesend() method, which is used to place the orders in the Monerium system.

Thesend(safeMoneriumOrder) you can access as a property in theSafeMoneriumClient instance method takes an order to be placed:

SafeMoneriumOrder { safeAddress : string amount : string currency : Currency counterpart : Counterpart memo : string }

And it will do the following:

1. Creates aredeem
2. order with the correct format for the Monerium SDK to understand
3. Place the order to Monerium
4. Propose asignMessage
5. transaction to the Safe services with the required Monerium message to be signed and executed. Monerium systems are aware that the Safe is a multisig wallet and will wait for this order to be confirmed and executed before carrying out the order in their systems.

Caveats

- The order we use internally in the SDK for evaluating theredirectUrl
- ,authCode
- andrefreshToken
- is important. Each property opens a different flow with Monerium and we evaluate the presence of theauthCode
- , then therefreshToken
- andredirectUrl
- as default. Have this in mind if you use all of them together in your app

subscribe(event, handler)

You can subscribe toorder status changes(opens in a new tab)through the Monerium API.

Parameters

- event
  - The event you want to subscribe to. You can choose between one of the following:

MoneriumEvent { placed =

'placed' , pending =

'pending' , processed =

'processed' , rejected =

'rejected' }

- handler
  - The handler function that will be called when the event is triggered.

unsubscribe(event, handler)

Allow to unsubscribe to authentication state changes.

Parameters

- event
  - The event you want to unsubscribe to.
- handler
  - The handler function that will be called when the event is triggered.

close()

Theclose method will clean up the socket, subscriptions and browser storage.

Usage

Instantiate the class and call theinit method when the page or component are loaded, followed by theopen(options) method when you want to start the interaction.

Theopen method starts the interaction with the pack and returns the Monerium SDK client enhanced with Safe specific methods.

// Instantiate and initialize the pack const

moneriumPack

=

new

MoneriumPack (moneriumConfig) moneriumPack .init ({ protocolKit })

// Open const

safeMoneriumClient

=

await

moneriumPack .open (moneriumPackOpenOptions)

// Subscribe to events const

handler

= (event) => {} moneriumPack .subscribe (MoneriumEvent .placed , handler) moneriumPack .unsubscribe (MoneriumEvent .processed , handler)

// Close await

moneriumPack .close ()

StripePack Relay Kit

Was this page helpful?

Report issue