

# MoneriumPack

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

## Install dependencies

To use the `MoneriumPack` , 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

The `MoneriumPack` class enables the use of Monerium services with Safe. To use it, create an instance of the pack and pass it to the `SafeOnrampKit` `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' }
```

The `clientId` is the secret representing the "Authorization Code Flow" for your Monerium account. To get your `clientId` :

1. Log in to [your account\(opens in a new tab\)](#)
2. .
3. Create a [new application\(opens in a new tab\)](#)
4. .

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

The `environment` is the environment for the Monerium SDK. You can choose between `production` and `sandbox` .

The production 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 Monerium [sandbox services \(opens in a new tab\)](#) and no KYC is required. Fake money will be used.

**Caveats** You should always call the `init()` method afterwards before interacting with the pack.

## **init(moneriumInitOptions)**

The `init` method initializes the Monerium SDK and the Safe services by creating a new instance of the [SafeMoneriumClient \(opens in a new tab\)](#) class. This class extends the [MoneriumClient \(opens in a new tab\)](#) class from the Monerium SDK and adds extra features to use it with the Safe services.

### Parameters

The `MoneriumInitOptions` options to be passed to the `init` method are:

`MoneriumInitOptions { protocolKit : Safe }`

- `protocolKit`
- - To use the `MoneriumPack`
- , you need to add Protocol Kit as a dependency for your project and create an instance of the [Safe \(opens in a new tab\)](#)
- class.

## **open(moneriumOpenOptions)**

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

### Parameters

The `MoneriumOpenOptions` options to be passed to the `open` method are:

`MoneriumOpenOptions { initiateAuthFlow ? : boolean }`

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

Take a look to [the example \(opens in a new tab\)](#) for more information.

### Returns

A `SafeMoneriumClient` 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 SDK [documentation \(opens in a new tab\)](#) .

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

The `send(safeMoneriumOrder)` you can access as a property in the `SafeMoneriumClient` 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 a `redeem`
2. order with the correct format for the Monerium SDK to understand
3. Place the order to Monerium
4. Propose a `signMessage`
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 the `redirectUrl`
- `,authCode`
- `andrefreshToken`
- is important. Each property opens a different flow with Monerium and we evaluate the presence of the `authCode`
- `, 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 to [order 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()

The `close` method will clean up the socket, subscriptions and browser storage.

## Usage

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

The `open` 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](#)