# Integration with Stripe

TheStripe Crypto Onramp service(opens in a new tab) allows individuals to securely purchase cryptocurrencies from your application.

## What are we going to learn?

This guide demonstrates how to use theStripePack as part of theOnrampKit (opens in a new tab) and incorporate it into your web application.

We are going to learn how to render the Stripe widget into your page. This widget allows the use your own Ethereum address for onramping cryptocurrencies. AsStripe API(opens in a new tab) usagerequires a server(opens in a new tab) to start the interaction with their services, we will also be using apre-deployed server(opens in a new tab) and providing a public key for testing purposes.

## Prerequisites

1. Node.js and npm(opens in a new tab)
2. Stripe account to get your own public and private keys(opens in a new tab)
3. A deployed server (example(opens in a new tab)
4. ) for communicating with Stripe APIs.
5. We are providing both the public key and the server for testing purposes but you must use your own public key and server in production.

## Install dependencies

yarn

add

@safe-global/onramp-kit

@stripe/stripe-js

@stripe/crypto

## Using theStripePack

in your Web App

TheStripePack can be used with any frontend framework like React, Vue, Angular or even plain HTML and JavaScript. In this example, we are using it with plain JavaScript.

1. Load the application and initialize theStripePack
2. using the following snippet:

const

stripePack

=

new

StripePack ({ // Get public key from Stripe: https://dashboard.stripe.com/register stripePublicKey : 'pk_test_51MZbmZKSn9ArdBimSyl5i8DqfcnlhyhJHD8bF2wKrGkpvNWyPvBAYtE211oHda0X3Ea1n4e9J9nh2JkpC7Sxm5a200Ug9ijfoO' , // Deploy your own server: https://github.com/5afe/aa-stripe-service onRampBackendUrl :

'https://aa-stripe.safe.global' }) await

stripePack .init ()

1. Add a container in any place in your web page. We choose to add adiv
2. container with an id ofstripe-root
3. :

< div

# id

"stripe-root"

</ div

1. Render the Stripe widget inside the container by providing the CSS selector to theelement

2. prop in theopen()
3. method:

// See options for using the StripePack open method in: // https://stripe.com/docs/crypto/using-the-api const

sessionData

=

await

stripePack .open ({ element :

'#stripe-root' ,

// Can be any CSS selector theme :

'light'

// light | dark // Optional, if you want to use a specific created session // --- // sessionId: 'cos_1Mei3cKSn9ArdBimJhkCt1XC', // Optional, if you want to specify default options // --- // defaultOptions: { // transaction_details: { // wallet_address: walletAddress, // lock_wallet_address: true // supported_destination_networks: ['ethereum', 'polygon'], // supported_destination_currencies: ['usdc'], // }, // customer_information: { // email: 'john@doe.com' // } })

Make sure you include theelement . Otherwise, you may get the following error:

You can also specify the default options for the widget. For example, you can specify the default wallet address, supported destination networks, and supported destination currencies. See theStripe API documentation(opens in a new tab) for more details. The default options you specify using theopen method will be passed through the Stripe API when using our provided server. When you create your own one (you need to do it on your production apps) you should do something similar.

1. Listening to events is important for understanding what's happening around. It helps us to create a proper UI in our web page.

Check theStripe frontend events(opens in a new tab) for the list of available events.

const

uiLoadedHandler

= () => { console .log ( 'UI loaded' ) }

const

sessionUpdatedHandler

= (e) => { console .log ( 'Session Updated' ,

e .payload) }

stripePack .subscribe ( 'onramp_ui_loaded' , uiLoadedHandler) stripePack .subscribe ( 'onramp_session_updated' , sessionUpdatedHandler)

...

stripePack .unsubscribe ( 'onramp_ui_loaded' , uiLoadedHandler) stripePack .unsubscribe ( 'onramp_session_updated' , sessionUpdatedHandler)

## Testing my dapp containing the Stripe widget

In production, each customer should pass an individual KYC process but probably you want to test your application before ð. You can use the following test data for bypass the KYC process while intest mode(opens in a new tab) .

Field Value Description Email 8404.john.smith@example.com Use any test or fake emails Phone Number +18004444444 Use +18004444444 for phone number OTP Verification Code 000000 Use 000000 for the OTP verification code First Name John Use any first name Last Name Verified Use Verified for the last name Birthday 01/01/1901 Use 01/01/1901 for successful identity verification Identification Type Social Security Number Select Social Security Number for identification type Identification Number 000000000 Enter 000000000 to fill the identification number field Country United States Select United States for country Address Line 1 address_full_match Use address_full_match for successful identity verification City Seattle Use Seattle for city State Washington Select Washington for state Zip Code 12345 Use 12345 for zip code Test Credit Card Number 4242424242424242 Use test credit card 4242424242424242 Expiration Date 12/24 Use future expiration date 12/24 CVC 123 Use any CVC 123 Billing Zip Code 12345 Use any zip code 12345 for billing

## Onramp Kit KYC test data - Examples

## StripePack complete React example

Check a complete example (opens in a new tab) in the safe-core-sdk repository. Follow the steps in the README.md (opens in a new tab) to run the example and configure the environment variables (VITE_MONERIUM_CLIENT_ID isn't necessary) for the pack following the .env.sample (opens in a new tab) .

Onramp Kit Monerium

Was this page helpful?

Report issue