SDK Integration

In this section we'll integrate the Biconomy SDK and create a Smart Account that will be used for all of our dApp interactions.

Remove unnecessary code

Let's get started with working on theindex.tsx file.

In the jsx of the file let's remove all elements of the page except for the Head tags and Main tags. Your Component should look like this:

export

default

function

Home ( )

{ return

(<

  < Head

  < title

  Session Keys < / title

  < meta name = "description" content = "Build a dApp powered by session keys"

/

  < / Head

  < main className = { styles . main }

< / main

  < /

) } Notice I changed my title and description, feel free to do that. On the main tags I added a className forstyles.main to get all my content the centered look.

Imports and Smart Account Config

Lets update the imports that we will need for this to work:

import

Head

from

"next/head" ; import

styles

from

"@/styles/Home.module.css" ; import

{ useState }

from

"react" ; import

{ createSmartAccountClient , BiconomySmartAccountV2 } from

"@biconomy/account" ; import

{ ethers }

from

"ethers" ; In the Home component add the following state variables:

const

[address , setAddress]

=

useState < string

    ("") ; const

[loading , setLoading]

=

useState < boolean

    (false) ; const

[smartAccount , setSmartAccount]

=

useState < BiconomySmartAccountV2 |

null

    (null ,) ; const

[provider , setProvider]

=

useState < ethers . providers . Provider |

null

    (null ,) ; We're going to track the address, smartAccount, and provider as well as a loading state for this component and save them into state so they can be used in child components.

Lets also get the bundlerUrl and paymasterApiKey from the dashboard.

const

bundlerUrl :

"https://bundler.biconomy.io/api/v2/80001/nJPK7B3ru.dd7f7861-190d-41bd-af80-6877f74b8f44" , const paymasterApiKey =

"paymasterApiKey" ; We will need the bundler however the paymaster is optional in this case. If you want to sponsor transactions later or allow users to pay for gas in erc20 tokens make sure to set this up but for the purposes of this tutorial we will not be sponsoring any transactions. The paymaster URL here will only work for tutorial contracts.

Connect Wallet and Create Smart Account

Let's create a connect button that connects to an EOA and launches a new smart account. Make sure you have an in browser wallet such as Metamask installed for this part of the tutorial.

Create a connect function:

const

connect

=

async

( )

=>

{ // @ts-ignore const

{ ethereum }

=

window ; try

{ setLoading (true) ; const provider =

new

ethers . providers . Web3Provider (ethereum) ; await provider . send ("eth_requestAccounts" ,

[ ]) ; setProvider (provider) ; const signer = provider . getSigner ( ) ; let biconomySmartAccount =

await

createSmartAccountClient ({ signer , bundlerUrl , biconomyPaymasterApiKey : paymasterApiKey , }) ; setAddress (await biconomySmartAccount . getAccountAddress ( )) ; setSmartAccount (biconomySmartAccount) ; setLoading (false) ; }

catch

(error)

{ console . error (error) ; } } ; This function takes care of the following:

- Get the Ethereum object injected by a wallet from the window
- Set loading state to true
- Create a provider and signer using ethers after getting permission from user
- We set provider in state
- Initialize the BiconomySmartAccountV2 by using the createSmartAccountClient method
- Initialize the smart account and save the address and instance of smart account to state and finally set loading state back to false.

Let's add some items to our component within the main tags:

< h1

    Session

Keys

Demo < / h1

    < h2

    Connect and transfer ERC20 tokens without signing on each transfer < / h2

    { ! loading &&

! address &&

< button onClick = { connect } className = { styles . connect }

    Connect to Web3 < / button

    } { loading &&

< p

    Loading

Smart

Account ... < / p

    } { address &&

< h2

Smart

Account :

{ address } < / h2

} Now if we hit connect we will be able to view our Smart Account address owned by our EOA.

You have now integrated the SDK succesfully, in the next section we will set up our session key and execute an ERC20 transfer from our component. Expand the code below to see what your component should look like now:

Expand for Code import

Head

from

"next/head" ; import

styles

from

"@/styles/Home.module.css" ; import

{ useState }

from

"react" ; import

{ createSmartAccountClient , BiconomySmartAccountV2 }

from

"@biconomy/account" ; import

{ ethers }

from

"ethers" ;

export

default

function

Home ( )

{ const

[address , setAddress]

= useState < string

("") const

[loading , setLoading]

= useState < boolean

(false) ; const

[smartAccount , setSmartAccount]

= useState < BiconomySmartAccountV2

|

null

```
    ( null ) ; const
[ provider , setProvider ]
= useState < ethers . providers . Provider
|
null
    ( null )
const
bundlerUrl :
"https://bundler.biconomy.io/api/v2/80001/nJPK7B3ru.dd7f7861-190d-41bd-af80-6877f74b8f44" , const paymasterApiKey =
"paymasterApiKey" ;
const
connect
=
async
( )
=>
{ // @ts-ignore const
{ ethereum }
=
window ; try
{ setLoading ( true ) const provider =
new
ethers . providers . Web3Provider ( ethereum ) await provider . send ( "eth_requestAccounts" ,
[ ] ) ; const signer = provider . getSigner ( ) ; setProvider ( provider ) let biconomySmartAccount =
await
createSmartAccountClient ( { signer , bundlerUrl , biconomyPaymasterApiKey : paymasterApiKey , } ) setAddress ( await
biconomySmartAccount . getAccountAddress ( ) ) setSmartAccount ( biconomySmartAccount ) setLoading ( false ) }
catch
( error )
{ console . error ( error ) ; } } ;
return
( <
    < Head
    < title
    Session
Keys < / title
    < meta name = "description" content = "Build a dApp powered by session keys"
/
```

```
        < / Head
        < main className = { styles . main }
        < h1
        Session
Keys
Demo < / h1
        < h2
        Connect and transfer ERC20 tokens without signing on each transfer < / h2
        { ! loading &&
! address &&
< button onClick = { connect } className = { styles . connect }
        Connect to Web3 < / button
        } { loading &&
< p
        Loading
Smart
Account ... < / p
        } { address &&
< h2
        Smart
Account :
{ address } < / h2
        } < / main
        < /
        ) }
```
Previous Initialize Frontend Next Create Session