Quickstart: React

@biconomy/use-aa •Docs

info Clone thecomplete quick start here Biconomy has two main packages:@biconomy/useAA which is designed forReact projects, and@biconomy/biconomy-client-sdk , which is our core SDK that can be used in any JavaScript environment.

If you are building a React project, we recommend starting with@biconomy/useAA (which is the focus of the following tutorial). You can always drop down to the underlying Core SDK if required.

1. Create aNext.js

project

npx create-next-app@latest

2. Install dependencies

- bun
- npm
- yarn
- pnpm

bun add viem wagmi @tanstack/react-query @biconomy/account @biconomy/use-aa npm install viem wagmi @tanstack/react-query @biconomy/account @biconomy/use-aa yarn add viem wagmi @tanstack/react-query @biconomy/account @biconomy/use-aa pnpm add viem wagmi @tanstack/react-query @biconomy/account @biconomy/use-aa

3. Add your Providers

Create your relevant providers for each of your dependencies. You will need to retrieve paymaster and bundler details from thebiconomy dashboard .

"use client" ; import

{ http }

from

"viem" ; import

{ BiconomyProvider }

from

"@biconomy/use-aa" ; import

{ polygonAmoy }

from

"viem/chains" ; import

{ WagmiProvider , createConfig }

from

"wagmi" ; import

{ QueryClient , QueryClientProvider }

from

"@tanstack/react-query" ;

export

default

function

```
Providers ( { children } :
{ children : React . ReactNode } )
{ const biconomyPaymasterApiKey = process . env . NEXT_PUBLIC_PAYMASTER_API_KEY
||
"" ; const bundlerUrl = process . env . NEXT_PUBLIC_BUNDLER_URL
||
"" ;
const config =
createConfig ( { chains :
[ polygonAmoy ] , transports :
{ [ polygonAmoy . id ] :
http ( ) , } , } ) ;
const queryClient =
new
QueryClient ( ) ;
return
( < WagmiProvider config = { config }
        < QueryClientProvider client = { queryClient }
        < BiconomyProvider config = { { biconomyPaymasterApiKey , bundlerUrl , } } queryClient = { queryClient }
        { children } < / BiconomyProvider
        < / QueryClientProvider
        < / WagmiProvider
        ) ; }
```

4. Send a Sponsored Transaction

4a. Get the smart account address for the connected user

```
const
{ smartAccountAddress }
=
useSmartAccount ( ) ;
```

4b. Build the mintTx

```
const
mintNftTx
=
( )
=> mutate ( { transactions :
{ to :
"0x1758f42Af7026fBbB559Dc60EcE0De3ef81f665e" , data :
```

```
encodeFunctionData ( { abi :

parseAbi ( [ "function safeMint(address _to)" ] ) , functionName :

"safeMint" , args :

[ smartAccountAddress ] , } ) , } , } ) ;
```

4c. Send the sponsored transaction and wire the 'wait' hook

```
const

{ mutate , data : userOpResponse , error , isPending , }

=

useSendSponsoredTransaction ( ) ;

const

{ isLoading : waitIsLoading , isSuccess : waitIsSuccess , error : waitError , data : waitData , }

=

useUserOpWait ( userOpResponse ) ;

return

( < div

    < button className = " bg-orange-500 hover:bg-orange-700 text-white font-bold py-2 px-4 rounded cursor-
    pointer " type = " button " onClick = { mintNftTx }

    { waitIsLoading || isPending ?

"Executing..."

:

"Mint an NFT" } </ button

    { ( error || waitError )

&&

< div

    ERROR </ div

    } </ div

    ) ;
```

4d. Handle the success state

```
useEffect ( ( )

=>

{ if

( waitData ?. success ===

"true" )

{ console . log ( waitData ?. receipt ?. transactionHash ) ; } } ,

[ waitData ] ) ;
```

Next Steps

Congratulations! You have sent your first sponsored transaction with Biconomy. You can now explore specific methodhere

. Previous Overview Next Quickstart: Core SDK