Initialize Smart Account

In this step we'll set up our node js script to create or display a smart account in our command prompt.

info This tutorial has a setup step in the previous sectionEnvironment Setup

Initialization

Let’s import our bundler package, and providers from the ethers package:

import

{ IBundler , Bundler }

from

"@biconomy/bundler" ; import

{

DEFAULT_ENTRYPOINT_ADDRESS

}

from

"@biconomy/account" ; import

{ providers }

from

"ethers" ; import

{ ChainId }

from

"@biconomy/core-types" ; IBundler is the typing for the Bundler class that we will create a new instance of.

Initial Configuration

const bundler : IBundler =

new

Bundler ({ bundlerUrl : "https://bundler.biconomy.io/api/v2/80001/nJPK7B3ru.dd7f7861-190d-41bd-af80-6877f74b8f44" , chainId : ChainId . POLYGON_MUMBAI , entryPointAddress :

DEFAULT_ENTRYPOINT_ADDRESS , }) ; * Now we create an instance of our bundler with the following:* a bundler url which you can retrieve from the Biconomy Dashboard * * chain ID, in this case we’re using Polygon Mumbai * * and default entry point address imported from the account package

import

{ BiconomySmartAccount , BiconomySmartAccountConfig , DEFAULT_ENTRYPOINT_ADDRESS , }

from

"@biconomy/account" ; import

{ Wallet , providers , ethers }

from

"ethers" ; Update your import from the account package to also include BiconomySmartAccount and BiconomySmartAccountConfig and also import Wallet, providers, and ethers from the ethers package.

const provider =

new

providers . JsonRpcProvider ("https://rpc.ankr.com/polygon_mumbai" ,) ; const wallet =

new

Wallet (process . env . PRIVATE_KEY

||

"" , provider) ; * We create a provider using a public RPC provider endpoint from ankr, feel free to use any service here such as Infura or Alchemy if you wish. We encourage you to use a private RPC endpoint for better efficiency in userOp creation. * Next we create an instance of the wallet associated to our Private key.

Now lets sup our paymaster. Update your imports to contain the following values:

import

{ IPaymaster , BiconomyPaymaster , IHybridPaymaster , PaymasterMode , SponsorUserOperationDto , }

from

"@biconomy/paymaster" ; We'll need these to help us execute our gasless transaction. The first thing we want to do is create an instance of our paymaster:

const paymaster : IPaymaster =

new

BiconomyPaymaster ({ paymasterUrl : "https://paymaster.biconomy.io/api/v1/80001/Tpk8nuCUd.70bd3a7f-a368-4e5a-af14-80c7f1fcda1a" , }) ; info Note that using the above paymaster URL will only work on Polygon Mumbai network and will only allow sponsored transactions from the contract address mentioned at the start of this tutorial. If you would like to learn how to use our dashboard to get your own paymaster url on any of our supported chains make sure to check out ourDashboard Documentation We now need an object that will hold the configuration values for our Smart Account.

const biconomySmartAccountConfig : BiconomySmartAccountConfig =

{ signer : wallet , chainId : ChainId . POLYGON_MUMBAI , bundler : bundler , paymaster : paymaster , } ; Now we can use this configuration to get the users Smart Account or create one if they don't already have it:

async

function

createAccount ( )

{ let biconomySmartAccount =

new

BiconomySmartAccount (biconomySmartAccountConfig ,) ; biconomySmartAccount =

await biconomySmartAccount . init ( ) ; console . log ("owner: " , biconomySmartAccount . owner) ; console . log ("address: " ,

await biconomySmartAccount . getSmartAccountAddress ( )) ; return biconomySmartAccount ; }

createAccount ( ) ; After running this script, your command prompt will display the owner of the smart account (the address linked to your private key) and the address of your smart account. In the next section, we'll execute our first gasless transaction by minting an NFT. Previous Environment Set up Next Creating Gasless Transactions