Creating Gasless Transactions

In this guide, we will walk through creating a basic Node.js script using TypeScript that allows user to mint an NFT without paying for any Gas.

This tutorial has a two other steps in the previous sections:Environment Setup andInitializing Account This tutorial will be done on the Polygon Mumbai Network and the smart contract for the NFT mint is availablehere . The address for the contract is 0x1758f42Af7026fBbB559Dc60EcE0De3ef81f665e.

Building the NFT Mint Function

Now it's time to construct and execute our NFT Minting function:

async

function

mintNFT ( )

{ } All of the code from this point forward will be code we add to the mintNFT function defined above. Let's start with creating an interface for our NFT contract.

const nftInterface =

new

ethers . utils . Interface (["function safeMint(address _to)" ,]) ; Now we'll need to collect the data needed for building our userOp, we'll do that using the ethers encodeFunctionData method:

const data = nftInterface . encodeFunctionData ("safeMint" ,

[address]) ;

Constructing the Userop

Let's declare a variable for our NFT contract address and start constructing the first part of our transaction:

const nftAddress =

"0x1758f42Af7026fBbB559Dc60EcE0De3ef81f665e" ;

const transaction =

{ to : nftAddress , data : data , } ; We will now start building out the partial userOp for this transaction:

let partialUserOp =

await smartAccount . buildUserOp ([transaction]) ;

Paymaster Service Data response

Now we need to construct the paymasterServiceData field of our userOp to make sure that our Paymaster is used to sponsor the transaction. We start by defining the following variables

const biconomyPaymaster = smartAccount . paymaster as IHybridPaymaster < SponsorUserOperationDto

;

let paymasterServiceData : SponsorUserOperationDto =

{ mode : PaymasterMode . SPONSORED , } ; With this done let's get the paymaster and Data response from our paymaster and add it to our userOp.

try

{ const paymasterAndDataResponse =

await biconomyPaymaster . getPaymasterAndData (partialUserOp , paymasterServiceData ,) ; partialUserOp . paymasterAndData = paymasterAndDataResponse . paymasterAndData ; }

catch

(e)

{ console . log ("error received " , e) ; }

Execute the UserOp with Paymaster

Finally let's go ahead and mint this NFT!

try

{ const userOpResponse =

await smartAccount . sendUserOp (partialUserOp) ; const transactionDetails =

await userOpResponse . wait ( ) ; console . log (ransactionDetails: https://mumbai.polygonscan.com/tx/ { transactionDetails . receipt . transactionHash } ,) ; console . log (view minted nfts for smart account: https://testnets.opensea.io/ { address } ,) ; }

catch

(e)

{ console . log ("error received " , e) ; } In this final try/catch block we send the user op and log out the transaction details into a nicely formatted polygonscan link as well as a formatted opensea link to quickly view NFT's minted by our smart account.

Click to view final code import

{ config }

from

"dotenv" ; import

{ IBundler , Bundler }

from

"@biconomy/bundler" ; import

{ BiconomySmartAccount , BiconomySmartAccountConfig , DEFAULT_ENTRYPOINT_ADDRESS , }

from

"@biconomy/account" ; import

{ Wallet , providers , ethers }

from

"ethers" ; import

{ ChainId }

from

"@biconomy/core-types" ; import

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

from

"@biconomy/paymaster" ;

config ( ) ;

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 , } ) ;

const paymaster : IPaymaster =

new

BiconomyPaymaster ( { paymasterUrl : "https://paymaster.biconomy.io/api/v1/80001/Tpk8nuCUd.70bd3a7f-a368-4e5a-af14-80c7f1fcda1a" , } ) ;

const provider =

new

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

new

Wallet ( process . env . PRIVATE_KEY

||

"" , provider ) ;

const biconomySmartAccountConfig : BiconomySmartAccountConfig =

{ signer : wallet , chainId : ChainId . POLYGON_MUMBAI , bundler : bundler , paymaster : paymaster , } ;

let smartAccount : BiconomySmartAccount ; let address :

string ;

async

function

createAccount ( )

{ console . log ( "creating address" ) ; let biconomySmartAccount =

new

BiconomySmartAccount ( biconomySmartAccountConfig , ) ; biconomySmartAccount =

await biconomySmartAccount . init ( ) ; address =

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

async

function

mintNFT ( )

{ await

createAccount ( ) ; const nftInterface =

new

ethers . utils . Interface ( [ "function safeMint(address _to)" , ] ) ;

const data = nftInterface . encodeFunctionData ( "safeMint" ,

[ address ] ) ;

const nftAddress =

"0x1758f42Af7026fBbB559Dc60EcE0De3ef81f665e" ;

const transaction =

{ to : nftAddress , data : data , } ;
```

```
console . log ( "creating nft mint userop" ) ; let partialUserOp =

await smartAccount . buildUserOp ( [ transaction ] ) ;

const biconomyPaymaster = smartAccount . paymaster as IHybridPaymaster < SponsorUserOperationDto

        ;

let paymasterServiceData : SponsorUserOperationDto =

{ mode : PaymasterMode . SPONSORED , } ; console . log ( "getting paymaster and data" ) ; try

{ const paymasterAndDataResponse = await biconomyPaymaster . getPaymasterAndData ( partialUserOp ,
paymasterServiceData , ) ; partialUserOp . paymasterAndData = paymasterAndDataResponse . paymasterAndData ; }

catch

( e )

{ console . log ( "error received " , e ) ; } console . log ( "sending userop" ) ; try

{ const userOpResponse =

await smartAccount . sendUserOp ( partialUserOp ) ; const transactionDetails =

await userOpResponse . wait ( ) ; console . log (ransactionDetails: https://mumbai.polygonscan.com/tx/ { transactionDetails . receipt .
transactionHash } , ) ; console . log (view minted nfts for smart account: https://testnets.opensea.io/ { address }  , ) ; }

catch

( e )

{ console . log ( "error received " , e ) ; } }
```

mintNFT ( ) ; Now that you've executed your first gasless transaction, let's edit this code in the next session to help us with batching multiple transactions together. Previous Initialize Smart Account Next Batching Multiple Transactions