# NFT Mint with ABI SVM

info In this instance we are not sponsoring the transaction please ensure that you send mumbai matic to your smart account to pay for gas. You will not be able to execute this code unless your smart account has some mumbai matic and mumbai usdc. Now the final part of this tutorial. Executing a NFT mint using a session!

## Create new component

Let's Create a new component calledNFTMint.tsx and place it in the components folder.

The imports and props will be as follows:

import React from

"react" ; import

{ ethers }

from

"ethers" ; import

{ BiconomySmartAccountV2 ,

DEFAULT_SESSION_KEY_MANAGER_MODULE , createSessionKeyManagerModule }

from

"@biconomy/account" import nftAbi from

"@/utils/nftAbi.json" import

{ toast , ToastContainer }

from

'react-toastify' ; import

'react-toastify/dist/ReactToastify.css' ;

interface

props

{ smartAccount : BiconomySmartAccountV2 ; provider : ethers . providers . Provider ; address :

string ; } Let's create the inital component:

const NFTMint : React . FC < props

=

( { smartAccount , provider , address } )

=>

{ return ( < button

Mint NFT < / button

) }

export

default NFTMint ; This is going to be a basic button that simply mints 1 NFT to a recipient. You can go ahead and import this component now into your Create Session component. It should look like this at the bottom of your Create Session component:

{ isSessionActive &&

( < NFTMint smartAccount = { smartAccount } provider = { provider } address = { address } /

) ; }

## Create the mintNft function

note Replace address : Make sure to replace the recipient address value (current 0xAddress). Let's create the function now to handle the transfer:

const

mintNft

=

async

( )

=>

{ if

( ! address ||

! smartAccount ||

! address )

{ alert ( "Please connect wallet first" ) ; return ; } try

{ toast . info ( 'Minting an NFT' ,

{ position :

"top-right" , autoClose :

15000 , hideProgressBar :

false , closeOnClick :

true , pauseOnHover :

true , draggable :

true , progress :

undefined , theme :

"dark" , } ) ; const abiSVMAddress =

"0x000006bC2eCdAe38113929293d241Cf252D91861" // get session key from local storage const sessionKeyPrivKey = window . localStorage . getItem ( "sessionPKey" ) ; console . log ( "sessionKeyPrivKey" , sessionKeyPrivKey ) ; if

( ! sessionKeyPrivKey )

{ alert ( "Session key not found please create session" ) ; return ; } const sessionSigner =

new

ethers . Wallet ( sessionKeyPrivKey , provider ) ; console . log ( "sessionSigner" , sessionSigner ) ;

// generate sessionModule const sessionModule =

await

createSessionKeyManagerModule ( { moduleAddress :

DEFAULT_SESSION_KEY_MANAGER_MODULE , smartAccountAddress : address , } ) ;

// set active module to sessionModule smartAccount = smartAccount . setActiveValidationModule ( sessionModule ) ;

const nftAddress =

"0xdd526eba63ef200ed95f0f0fb8993fe3e20a23d0" ; const recipient =

"0xAddress" ; const nftContract =

new

ethers . Contract ( nftAddress , nftAbi , provider ) ;

const

{ data }

=

await nftContract . populateTransaction . safeMint ( recipient ) ;

// generate tx data const tx =

{ to : nftAddress , data : data ! } ;

// This will build the tx into a user op and send it. let userOpResponse =

await smartAccount . sendTransaction ( tx ,

{ params :

{ sessionSigner : sessionSigner , sessionValidationModule : abiSVMAddress , } , } ) ;

console . log ( "userOpHash" , userOpResponse ) ; const

{ receipt }

=

await userOpResponse . wait ( 1 ) ; console . log ( "txHash" , receipt . transactionHash ) ; const polygonScanlink =

https://mumbai.polygonscan.com/tx/ { receipt . transactionHash } toast . success ( < a target = "_blank" href = { polygonScanlink }

Success Click to view transaction < / a

,

{ position :

"top-right" , autoClose :

18000 , hideProgressBar :

false , closeOnClick :

true , pauseOnHover :

true , draggable :

true , progress :

undefined , theme :

"dark" , } ) ; }

catch ( err :

any )

{ console . error ( err ) ; } } Let's break down this code:

if

( ! address ||

! smartAccount ||

! address )

{ alert ( "Please connect wallet first" ) ; return ; } First we check to make sure our props all exist.

toast . info ( "Minting an NFT" ,

{ position :

"top-right" , autoClose :

15000 , hideProgressBar :

false , closeOnClick :

true , pauseOnHover :

true , draggable :

true , progress :

undefined , theme :

"dark" , } ) ; We update the user that a mint nft is about to start

const abiSVMAddress =

"0x000006bC2eCdAe38113929293d241Cf252D91861" // get session key from local storage const sessionKeyPrivKey = window . localStorage . getItem ( "sessionPKey" ) ; console . log ( "sessionKeyPrivKey" , sessionKeyPrivKey ) ; if

( ! sessionKeyPrivKey )

{ alert ( "Session key not found please create session" ) ; return ; } const sessionSigner =

new

ethers . Wallet ( sessionKeyPrivKey , provider ) ; console . log ( "sessionSigner" , sessionSigner ) ; We specify the ABI SVM address and get the private key we stored in local storage and create a new session signer from it.

// generate sessionModule const sessionModule =

await

createSessionKeyManagerModule ( { moduleAddress :

DEFAULT_SESSION_KEY_MANAGER_MODULE , smartAccountAddress : address , } ) ;

// set active module to sessionModule smartAccount = smartAccount . setActiveValidationModule ( sessionModule ) ; Now we'll generate a session module using the Session Key Manager Module and then set the active validation module to be the session module. This updates the original configureation on the smart account.

const nftAddress =

"0xdd526eba63ef200ed95f0f0fb8993fe3e20a23d0" ; const recipient =

"0xAddress" ; const nftContract =

new

ethers . Contract ( nftAddress , nftAbi , provider ) ; We now create an instance of our NFT contract. Our contract has a safeMint function which requires just a recipient parameter.

const

{ data }

=

await nftContract . populateTransaction . safeMint ( recipient ) ; Now we will get raw transaction data for a nft mint to the recipient address we specified. Using any other recipient other than the one registered on the session key will result in an error.

const tx =

{ to : nftAddress , data : data } ;

// This will build the tx into a user op and send it. let userOpResponse =

await smartAccount . sendTransaction ( tx ,

{ params :

{ sessionSigner : sessionSigner , sessionValidationModule : abiSVMAddress , } , } ) ; Now we build the user op and send it for execution. The sendTransaction method will build the user op and send it to the Bundler.

console . log ( "userOpHash" , userOpResponse ) ; const

{ receipt }

=

await userOpResponse . wait ( 1 ) ; console . log ( "txHash" , receipt . transactionHash ) ; const polygonScanlink =

https://mumbai.polygonscan.com/tx/ { receipt . transactionHash } ; toast . success ( < a target = "_blank" href = { polygonScanlink }

Success Click to view transaction < / a

, { position :

"top-right" , autoClose :

18000 , hideProgressBar :

false , closeOnClick :

true , pauseOnHover :

true , draggable :

true , progress :

undefined , theme :

"dark" , } , ) ; Finally to give the user a succesful feedback we provide them with a link to the transaction once it has been executed.

View Complete Code import React from

"react" ; import

{ ethers }

from

"ethers" ; import

{ BiconomySmartAccountV2 ,

DEFAULT_SESSION_KEY_MANAGER_MODULE , createSessionKeyManagerModule }

from

"@biconomy/account" import nftAbi from

"@/utils/nftAbi.json" import

{ toast , ToastContainer }

from

'react-toastify' ; import

'react-toastify/dist/ReactToastify.css' ;

interface

props

{ smartAccount : BiconomySmartAccountV2 ; provider : ethers . providers . Provider ; address :

string ; }

const NFTMint : React . FC < props

=

( { smartAccount , provider , address } )

=>

{

const

mintNft

=

async

( )

=>

{ if

( ! address ||

! smartAccount ||

! address )

{ alert ( "Please connect wallet first" ) ; return ; } try

{ toast . info ( 'Minting an NFT' ,

{ position :

"top-right" , autoClose :

15000 , hideProgressBar :

false , closeOnClick :

true , pauseOnHover :

true , draggable :

true , progress :

undefined , theme :

"dark" , } ) ; const abiSVMAddress =

"0x000006bC2eCdAe38113929293d241Cf252D91861" // get session key from local storage const sessionKeyPrivKey = window . localStorage . getItem ( "sessionPKey" ) ; console . log ( "sessionKeyPrivKey" , sessionKeyPrivKey ) ; if

( ! sessionKeyPrivKey )

{ alert ( "Session key not found please create session" ) ; return ; } const sessionSigner =

new

ethers . Wallet ( sessionKeyPrivKey , provider ) ; console . log ( "sessionSigner" , sessionSigner ) ;

// generate sessionModule const sessionModule =

await

createSessionKeyManagerModule ( { moduleAddress :

DEFAULT_SESSION_KEY_MANAGER_MODULE , smartAccountAddress : address , } ) ;

// set active module to sessionModule smartAccount = smartAccount . setActiveValidationModule ( sessionModule ) ;

const nftAddress =

"0xdd526eba63ef200ed95f0f0fb8993fe3e20a23d0" ; const recipient =

"0xAddress" ; const nftContract =

new

ethers . Contract ( nftAddress , nftAbi , provider ) ;

const

{ data }

=

await nftContract . populateTransaction . safeMint ( recipient ) ;

// generate tx data const tx =

{ to : nftAddress , data : data ! } ;

// This will build the tx into a user op and send it. let userOpResponse =

await smartAccount . sendTransaction ( tx ,

{ params :

{ sessionSigner : sessionSigner , sessionValidationModule : abiSVMAddress , } , } ) ;

console . log ( "userOpHash" , userOpResponse ) ; const

{ receipt }

=

await userOpResponse . wait ( 1 ) ; console . log ( "txHash" , receipt . transactionHash ) ; const polygonScanlink =

https://mumbai.polygonscan.com/tx/ { receipt . transactionHash } toast . success ( < a target = "_blank" href = { polygonScanlink }

Success Click to view transaction < / a

,

{ position :

"top-right" , autoClose :

18000 , hideProgressBar :

false , closeOnClick :

true , pauseOnHover :

true , draggable :

true , progress :

undefined , theme :

"dark" , } ) ; }

catch ( err :

any )

{ console . error ( err ) ; } }

return ( < button onClick = { mintNft }

< div

< ToastContainer position = "top-right" autoClose = { 5000 } hideProgressBar = { false } newestOnTop = { false } closeOnClick rtl = { false } pauseOnFocusLoss draggable pauseOnHover theme = "dark" /

< button onClick = { mintNft }

Mint NFT < / button

< / div

< / button

) }

export

default NFTMint ; Running this code should now allow you to sign in using your EOA, create a session, and then mint an NFT without the need to sign any further transactions!

What's also remarkable is the ability to define your own rules according to the logic of your smart contract. Don't limit yourself to just minting NFTs, the possibilities are endless with ABI SVM! Previous Create Session Next Batched Session Tutorial